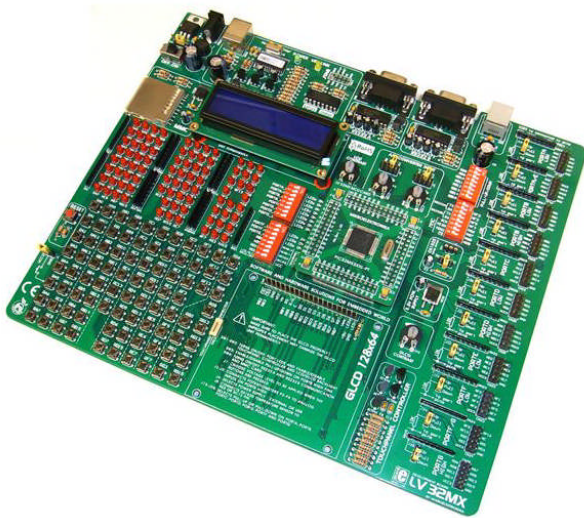
型番:PICD-32MX　　　　　　　　　　　　REV.1.00

USB2.0対応　PICライター搭載
PIC32MX開発用統合評価ボード

取扱説明書
お使いになる前にこの説明書をよくお読みの上正しくお使いください。

ボード本体

製品の概要

　PIC30MX開発用統合評価ボード[型番:PICD-32MX、以下PICD-32MXと記載]は、オンボードにUSB接続のPICライターを搭載し、最新の32ビットPIC、PIC32MXの開発に対応した統合開発ボードです。
　ボード上には、100ピンのMCUカード用ソケットが用意されており標準ではPIC32MX360F256L-80のMCUカードが付属しています。
　ボード上に搭載のPICライターは、最新のPIC32MXシリーズのデバイスに対応しており、基板上に装着したMCUカードに搭載のマイコンにプログラムを書き込みます。USB接続を行った場合、USBバスからPICD-32MXの電源を取りますので、別途電源を準備する必要がなく、パソコンと接続すればすぐに開発を始められます。

　ボードに搭載されている各周辺回路に接続されているピンは周辺回路に接続して利用するか又は周辺回路には接続しないで使用するかをディップスイッチやジャンパーソケットで選択できます。またI/Oピンはボード右側に実装されているヘッダピンから取り出すことができます。
　周辺回路やI/Oピンの状態を表示するLEDやタクトスイッチ等を使うことでPIC32MXの評価からアプリケーションの開発、実験が簡単に行えます。
　マイクロチップ社からの純正Cコンパイラー、MPLAB C32を利用することでCコンパイラーによる開発が可能となります。32ビットPICのPIC32MXの開発を簡単にはじめたい方におすすめの評価ボードです。

パッケージの内容

■同梱物
- ・PICD-32MXボード本体
- ・PIC32MX360F256L-80搭載MPUボード　（ボードに装着済み）
- ・16文字×2行液晶ディスプレイモジュール　（ボードに装着済み）
- ・USBケーブル
- ・CD-ROM
- ・マニュアル(本書)

ドライバのインストール及びパソコンとの接続

■デバイスドライバーのインストール
　PICD-32MXをパソコンと接続する前に、デバイスドライバーをパソコンにインストールします。(Windows98及びMEをご使用のお客様は本項は読み飛ばして、次の「パソコンと接続する」の項よりお読み下さい。)WindowsVista及びWindowsXPをご利用のお客様は下記の手順で、ドライバーをインストールします。

1　付属のCD-ROMをパソコンのCD-ROMドライブに挿入してエクスプローラー等で内容を表示します。

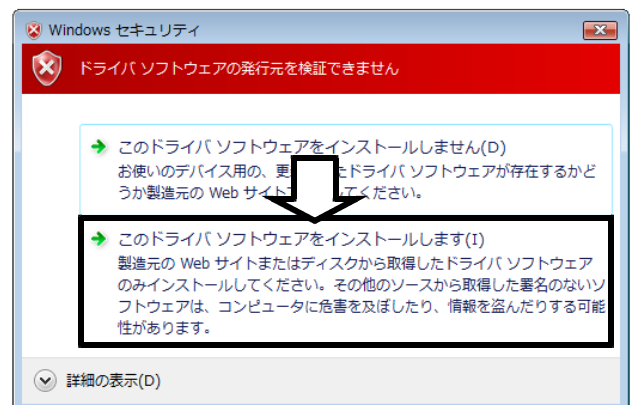
2　”USB Programmer”のフォルダを開きます。

3　”Driver”フォルダを開きます。

4　OSごとにフォルダがあります。ご使用のパソコンのOSにあったフォルダを開きます。(WindowsVista及びXP用には32ビット用と64ビット用がありますのでご注意ください。)

5　開くと実行ファイル、”USB18PRG～.exe”がありますのでダブルクリックして実行します。

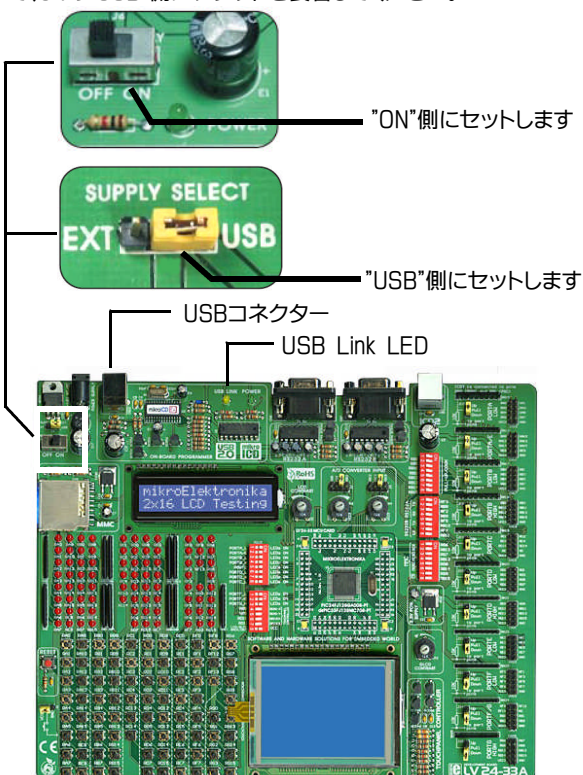
6　実行するとダイアログが表示されますので、そのまま”次へ”をクリックして続行します。

7　ドライバーのインストールが開始されます。
WindowsVista環境で下記のような警告ダイアログが表示された場合には、”このドライバソフトウエアをインストールします”をクリックして続行してください。

下図のようなメッセージが表示された場合には、"インストール"ボタンを押してください。

8　インストール作業が完了すると、下図のような画面になりますので、"完了"ボタンを押してインストールを完了します。
この時、"状態"のボックスに「使用できます」と表示されていることをご確認下さい。エラーが表示された場合には、ご使用OSの種類と、実行したデバイスドライバーの種類が一致しているかご確認下さい。

■パソコンと接続する

パソコンのUSBポートにPICD-32MXを接続します。

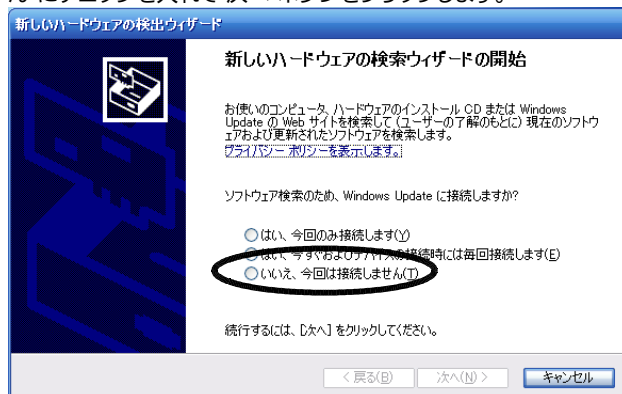
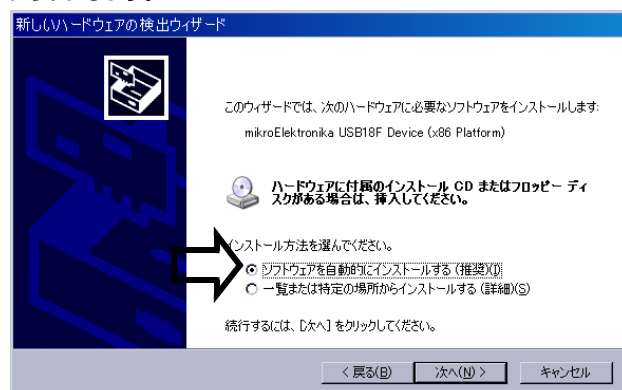
1　PICD-32MXの電源スイッチをONにします。
また、PICD-32MXの電源をUSBのバスパワーより給電しますので、J4の"USB"側にソケットを装着してください。

2　USBケーブルで、パソコンのUSBポートと、PICD-32MXを接続します。

※パソコンのUSBポートがUSB2.0対応の場合には高速書き込みが行えます。USB1.1ポートの場合には通常速度での書き込みとなります。

→接続するとPOWERの緑LEDが点灯します。
→新しいハードウエアの検出ウィザードが自動的に起動します。

以下OSによって操作方法が異なります。

■Windows2000, XPをご利用の場合------------------

3　下図のようなダイアログが表示されたら"いいえ、今回は接続しません"にチェックを入れて"次へ"ボタンをクリックします。

4　下記のようなダイアログが表示されますので、"ソフトウエアを自動的にインストールする(推奨)"にチェックをいれて、"次へ"ボタンをクリックします。

5　「ソフトウエアをインストールしています。お待ち下さい」というメッセージが表示されます。しばらくするとインストールが完了して下図のような表示となりますので"OK"ボタンをクリックして終了します。

■WindowsVistaをご利用の場合

3 WindowsVistaの場合には、PICD-32MXを接続すると自動的にデバイスドライバーのセットアップが開始されます。
あらかじめデバイスドライバーは先の手順にてインストールされていますので、WindowsVistaは自動的にデバイスドライバーを検出してインストールを完了します。

接続後しばらく待ちますと、下図のようなポップアップがタスクバーに表示されます。

上図のホップアップが表示されればインストールは完了です。

■Windows98(SE), MEをご利用の場合

3 自動的にデバイスが検出されます。
"新しいハードウエアの追加ウィザード"が表示されたら「次へ」を押して続行します。

4 次のダイアログでは,"使用中のデバイスに最適なドライバを検索する"にチェックを入れて,「次へ」をクリックします。

5 "検索場所の指定"にチェックを入れて,「参照」ボタンを押します。
ドライバの場所を指定するダイアログが表示されますのでCD-ROM内の"USB Programmer Software"フォルダ内の"Driver"フォルダにある"WIN98_ME"フォルダを指定してください。CD-ROMドライブがQドライブの場合にはディレクトリは下記の通りとなります。
Q:¥USB Programmer Softeare¥Driver¥WIN98_ME

指定したら「次へ」を押して続行します。
インストールが完了したら,「完了」ボタンを押して終了します。

～ここからは各OS共通の項目です～

PICD-32MXボード上の"USB Link"の黄LEDが点灯していることを確認してください。
黄LEDの点灯はPC側にPICD-32MXが正しく認識されていることを示します。

※"USB Link"のLEDが点灯していない場合には、正しくドライバーがインストールされていません。再度手順を確認の上、ドライバーをインストールし直しておためしください。

※デバイスマネージャーに"USB Devices by mikroElektronika"が追加されます。

WIndowsVista環境でご使用のお客様へ ユーザーアカウント制御無効設定のお願い

WindowsVista環境で、PICD-32MXをご使用の場合、WindowsVistaに搭載されたユーザーアカウント制御(UAC)機能がUSBポートへのアクセスを拒否することにより、下図のようなエラーメッセージが表示され書き込みができないという現象を確認しています。

この現象は、WindowsVistaにおいて標準で有効になっているユーザーアカウント制御機能を無効に設定することで回避できます。下記の方法で設定を変更して頂けますようお願い致します。

※なおこのエラーは、WindowsVista以外の環境でお使いのお客様には関係ありません。

■Windows Vistaのユーザーアカウント制御を無効に設定する

1 スタートボタンをクリックして、コントロールパネルを表示します。

2 コントロールパネルから"ユーザーアカウント"をダブルクリックします。"ユーザーアカウント"アイコンがない場合には、ウインドウ左上の"クラシック表示"をクリックして表示を変更します。

3 "ユーザーアカウント制御の有効化または無効化"のリンクをクリックします。

4 "ユーザーアカウント制御(UAC)を使ってコンピューターの保護に役立たせる"のチェックを外します。

5 "OK"ボタンを押して完了します。パソコンを再起動します。

MPLAB C32のダウンロード

PIC32用のCコンパイラーは、MPLAB C32というコンパイラーが、マイクロチップ社からリリースされています。製品版も販売しておりますが、Evaluation Versionという無償で利用できるものがあります。
Evaluation Versionは、最初の60日間は製品と同等の内容で動作しますが、期限が超過すると、最適化機能が固定されとなり、ファイルサイズが多少大きくなったり、実行速度が遅くなったりします。
ただし、その他の機能は製品版と全く同じで、大規模なプログラムでなければ実質このEvaluation Versionで十分開発が進められます。

MPLAB C32は、MPLABという統合開発環境の中で動作します。MPLABは付属のCD-ROMに収録されていますが、MPLAB C32はマイクロチップ社のWEBサイトにユーザー登録してからダウンロードすることになります。下記の手順でダウンロードしてください。

1 当方のダウンロードページにアクセスします。
http://www.microtechnica.net/manual/

2 項目20番にMPLAB C32のダウンロードサイトへのリンクがありますので、リンクをクリックします。マイクロチップ社のMPLAB C32のページが表示されます。

3 画面下の方に下がると"Download"という項目がありますので、その一覧から"MPLAB C Compiler for PIC32 v1.xx Evaluation Version"というリンクをクリックします。

Downloads
Sign in required to download content.

Title	Dat
C32 C Compiler Examples	4/27
MPLAB Assembler, Linker and Utilities for PIC32 MCUs User's Guide	5/27
MPLAB C Compiler for PIC32 v1.05 Evaluation Version	3/9/
MPLAB C Compiler for PIC32 v1.05 upgrade	3/9/
MPLAB C32 C Compiler User's Guide	10/

4 "Sing In"画面が表示されます。マイクロチップ社のアカウントをお持ちでしたらメールアドレスとパスワードを入力して"Sing In"をクリックします。
もしアカウントをお持ちでない場合には、新規登録をします。画面中央上部にある"register"というリンクをクリックします。

5 画面の指示に従い必要事項を入力します。
登録が正しく完了すると、入力したメールアドレスとパスワードによってサインインできるようになりますので、サイトにサインインします。

6 手順3の画面を表示して、"MPLAB C Compiler for PIC32 v1.xx Evaluation Version"というリンクをクリックし、ソフトウエアをダウンロードしてください。

ソフトウエアのインストール

開発に必要なソフトウエアをインストールします。
<u>なお、ソフトウエアをインストールする際には、必ずUSBポートからPICD-32MXを外した状態で行ってください。</u>

■MPLABのインストール
MPLABはPICマイコンのプログラム開発用の統合開発環境です。PICマイコンの開発元マイクロチップ社から提供されています。
MPLABがお手持ちのパソコンにインストールされていない場合には、インストールしてください。

1 CD-ROM内の"MPLAB"フォルダを開きます。
"SETUP.EXE"をダブルクリックして実行します。
"Next>"をクリックして続行します。

2 "License Agreement"のダイアログが表示されたら内容をよくご確認の上、同意する場合には"I accept the terms of the licence agreement"にチェックを入れて、"Next>"クリックして続行します。
なお同意しないとインストールは続行できません。

3 "setup type"の選択ダイアログではすべてのプログラムをインストールしますので"Complete"にチェックを入れて"Next>"をクリックします。

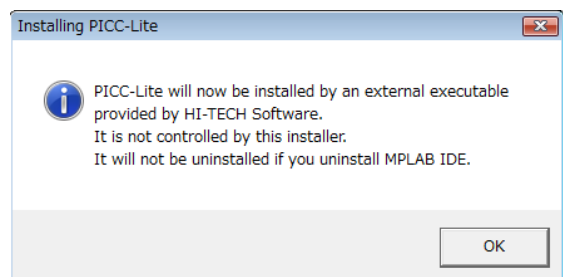
4 MPLABをインストールするディレクトリを指定します。
特に設定する必要がなければデフォルトのままにしておきます。
インストールするディレクトリを変更したい場合には,"Browse"ボタンを押してインストールディレクトリを設定します。
なお変更した場合には,インストールディレクトリをメモするなど忘れないようにしてください。
"Next>"をクリックして続行します。

5 以降のダイアログでは次のように進めます。

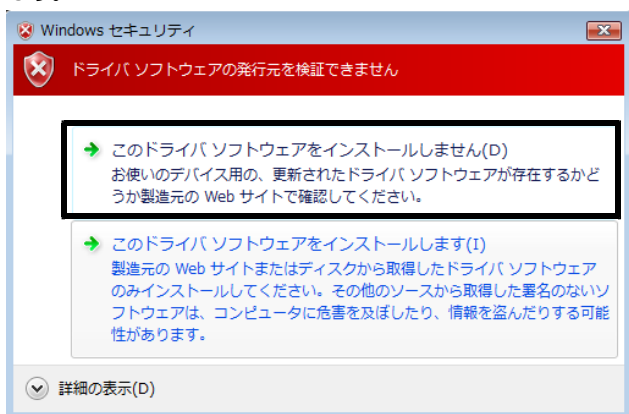
■Application Maestro License
→"I accept the terms of the license agreement"にチェックをいれて"Next>"をクリックします。

■MPLAB C32 License
→"I accept the terms of the license agreement"にチェックをいれて"Next>"をクリックします。

■Start copying files
→"Next>"を押します。
・・・インストールが開始されます。完了までしばらく待ちます。

6 インストールが完了すると、下図のようなダイアログが表示されます。MPLABバージョン8.00以降からはPICCコンパイラーのライトバージョンがMPLABに付属するようになったためです。
ここではOKボタンを押して続行します。

7　下図のようなPICC-Liteのインストール画面が表示されますので、ここでは"Cancel"ボタンを押してインストールをキャンセルします。

※PICC-Liteは、Hi-Tech社のPIC用Cコンパイラーです。インストールしていただいてもかまいませんが、本キットではこのソフトウエアは使用しません。

8　Windows Vista環境にインストールされている場合には、最後に下図のようなドライバーインストールに関するセキュリティ警告が表示されることがあります。
この場合には、"このドライバソフトウエアをインストールしません(D)"をクリックしてください。
ダイアログが消えたら"Finishボタンを押してインストールを完了します。

■MPLAB C32のインストール
　先ほどダウンロードしたPIC32用のCコンパイラー、MPLAB C32をインストールします。

1　ダウンロードした実行ファイルをダブルクリックして実行します。
インストーラーが起動します。

2　ウィザードが起動したら"Next"をクリックして続行します。
"License Agreement"が表示されますので、同意される場合には"I accept the terms of the license agreement"にチェックを入れて、"Next"ボタンを押します。

3　"License Key"の入力する画面となりますが、ここでは空欄のまま"Next"を押して続行します。

4　インストールディレクトリを指定します。このとき、必ず先にインストールしたMPLABがインストールされているディレクトリと同じ"Microchip"ディレクトリの下にインストールするよう設定してください。

通常はそのままでかまいませんが、MPLABをインストールしたとき、インストールディレクトリをデフォルト設定から変えていたり、Cドライブ以外のドライブにインストールした場合には、ここで設定を変更する必要があります。間違えないようご注意ください。

5　"Install"ボタンをクリックするとインストールが始まります。
画面の指示に従って、完了します。

■PICプログラマ用のソフトウエア
　PICへプログラムを書き込む際に使用するプログラマソフトウエアです。下記の手順でインストールを行います。

1　CD-ROM内の"USB Programmer Software"内にある"SETUP.EXE"をダブルクリックして実行します。インストーラーが起動しますので、"Next"をクリックして続行します。

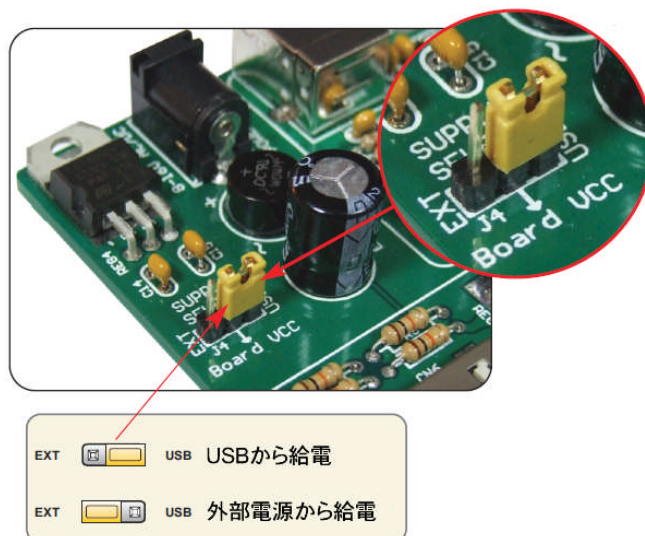
2　ライセンス許諾画面が表示されますので、同意される場合には、"I accept the terms in the License Agreement"の方にチェックを入れて、"Next"ボタンをクリックします。

3　次の画面はそのまま"Next"をクリックします。

4　インストールするディレクトリを指定します。
"Install"ボタンを押すとインストールが開始されます。

5　インストールが完了したら"Finish"ボタンを押して完了します。

電源の投入方法

　PICD-32MXは、USB接続にてUSBバスから電源の供給を受けることができます。この場合、別途電源を供給する必要はありません。
しかしPICD-32MX側で多くの電流(およそ350mA以上)を消費する予定がある場合にはUSBバスパワー給電では電力が不足する場合があります。この場合には別途外部からACアダプタにて電源を供給する必要があります。
　給電方法はジャンパーソケットJ4によって設定を行います。
なお工場出荷時の設定はUSBバスパワー給電の設定になっています。

■USBバスパワーから給電する場合
　PICD-32MXボード上の"SUPPLY SELECT"ジャンパーピンを「USB」側にショートします。

■ACアダプタから給電する場合
　PICD-32MXボード上の"SUPPLY SELECT"ジャンパーピンを「EXT」側にショートします。

※ACアダプタは,DC9V～12Vで500mA以上のものをご用意ください。径は2.1φです。(当方にて専用ACアダプタを販売中です)

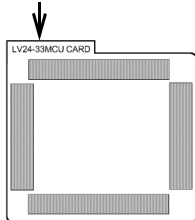
MCUカードの装着

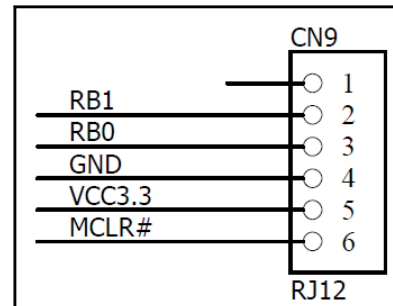
　お買い上げ時は、PIC32MX360F356L-80を搭載したMCUカードがすでに装着されています。
その他のデバイスを使用する場合には、MCUカードを交換します。交換は下記の手順で慎重に行ってください。

1 PICD-32MXの電源を切断します。

2 現在装着されているMCUカードをゆっくりと上に引き上げます。

3 MCUカードを装着します。
MCUカードには取り付け方向があります。取り付け方向を間違えるとボードやMCUカードが破損しますので下記のようにPICD-32MX基板上のシルク印刷に合わせて装着してください。

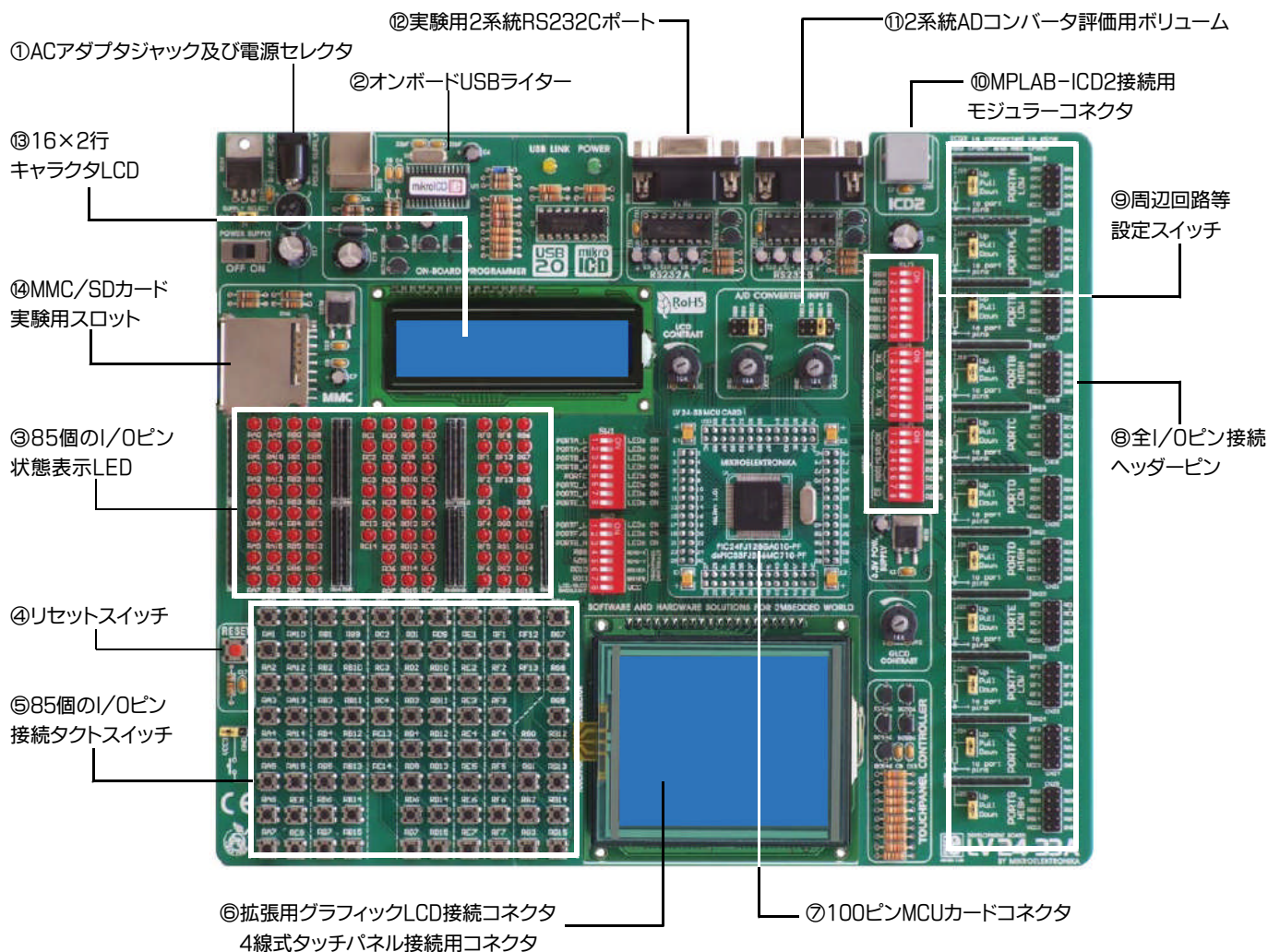
この突出している部分の方向を合わせて装着してください

MPLAB-ICD2を使用する場合

　PICD-32MXには、マイクロチップ社純正のICD、MPLAB-ICD2を接続するための6ピンモジュラーコネクターが付いています。ここにMPLAB-ICD2を接続することでPICD-32MXをターゲットボードとして使用することができます。配線は下図の通りです。

　ICD2を利用する場合には、RB1及びRB0はICD2に占有されますので外部回路を接続したり、プルアップやプルダウンをしないようにします。

■MPLAB-ICD2使用時のPICD-32MXの電源について
　ICD2用のモジュラーコネクタのVddピンは、PICD-32MXのVdd(+3.3V)と接続されています。MPLAB-ICD2接続時には、電源の取り扱い及びMPLAB-ICD2の電源設定を下記のように設定してください。設定を間違うと、機器を破損させる恐れがありますのでICD2を接続する前に必ず配線や設定が正しいか十分ご確認の上接続をしてください。

・PICD-32MXの電源は必ずACアダプタにて外部電源から給電する
・PICD-32MXのUSBはパソコンと接続しない
・MPLAB-ICD2は、"Target has own power supply"に設定する

以上の設定を正しく行ってご使用ください。
なお、マイクロチップ社のMPLAB-ICD3は仕様上ご使用頂けませんので、あらかじめご了承ください。

PICD-32MXの各部の説明

①ACアダプタジャック及び電源セレクトジャンパ

PICD-32MXの電源をACアダプタから給電する場合には、このジャックにDC9V〜12V出力で2.1φのACアダプタを接続します。センター極性は問いません。USBバスパワー給電では、およそ350mAまで使用できますがそれ以上の電流を消費するような場合にはACアダプタを使用してください。

給電方法はJ4の"SUPPLY SELECT"ジャンパーソケットで選択します。USBバスパワーから給電する場合はUSBと印刷された方に、ACアダプタから給電する場合はEXTと印刷された方にジャンパーソケットをセットします。

②オンボードUSBライター、USBコネクタ

オンボードUSB2.0マイコンライターのUSBポートです。

③85個I/Oピン状態表示LED群

PORTA(RA)〜PORTG(RG)の各ビットの状態を表示できるLED群です。RA・RB・RDは16ビット全ビットにLEDが接続されています。

RC・RE・RF・RGは、ユーザーが使用できるビットに対してLEDが接続されています。

該当のピンがHレベルになると点灯します。

LEDとマイコンのピンを接続するか、接続しないかの設定をディップスイッチSW1及びSW2にて設定できます。

スイッチの"LED's ON"と基板上に印刷された方にスイッチを設定すると、そのポートがLEDと接続されます。基本的に16ビットある各ポートのピンに対して上位8ビットと、下位8ビットの8ビット毎に分けて設定ができます。

なお、一部スイッチの設定が他のポートと兼用になっているものがありますので、基板上のシルク印刷を参考に設定をしてください。

マイコンのI/OピンにLEDが物理的に接続されていると、LED側に電流が流れ電圧降下が発生します。よって、ピンをI/Oピンとして使用する場合、LEDと接続されていると電圧降下により、Hレベル時にVcc電圧となりません。H-LにてVp-pをVcc-GND電圧で使用したい場合にはSW1、SW2の該当ポートのスイッチをOFF側に設定して、物理的にLEDからI/Oピンを切り離してご利用ください。

④リセットスイッチ

マイコンの$\overline{\text{MCLR}}$ピンと接続されており、スイッチを押すとPICマイコンにハードウエアリセットがかかります。

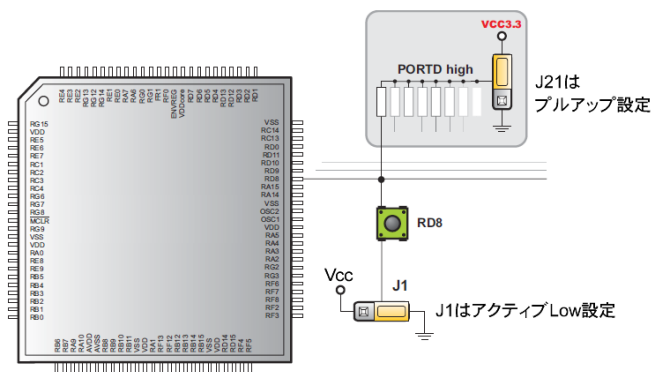
⑤85個I/Oピン接続タクトスイッチ群

J1のジャンパー設定によって、アクティブHigh又はアクティブLowが選択可能な85個のタクトスイッチです。

タクトスイッチ群の左側にあるJP1をVcc3と刻印された側に設定すると、アクティブHighに、GNDと刻印された側に設定すると、アクティブLowになります。

※アクティブHighは、ボタンを押した時に該当ピンがHレベルになり、アクティブLowは、ボタンを押した時に該当ピンがLレベルになる設定のことです。

各スイッチのプルアップ及びプルダウンは、各ポート8ビット単位(上位8ビット、下位8ビット)で設定できます。プルアップ/プルダウンの設定は、"⑧全I/O外部接続用ピン"の部分でジャンパー設定J15〜J25によって設定できます。

押しボタンスイッチの設定は、J1のアクティブLow/High設定と、J15〜J25のプルアップ/プルダウンの設定を適切に組み合わせる必要があります。J1をアクティブLow設定にした場合には、該当する押しボタンスイッチの接続されたポートは、プルアップしておく必要があります。下記に例を示します。

逆に、JP1をアクティブHigh設定にした場合には定常時がLowレベルになっている必要がありますので、JP15〜JP25は、プルダウン側にセットすることになります。

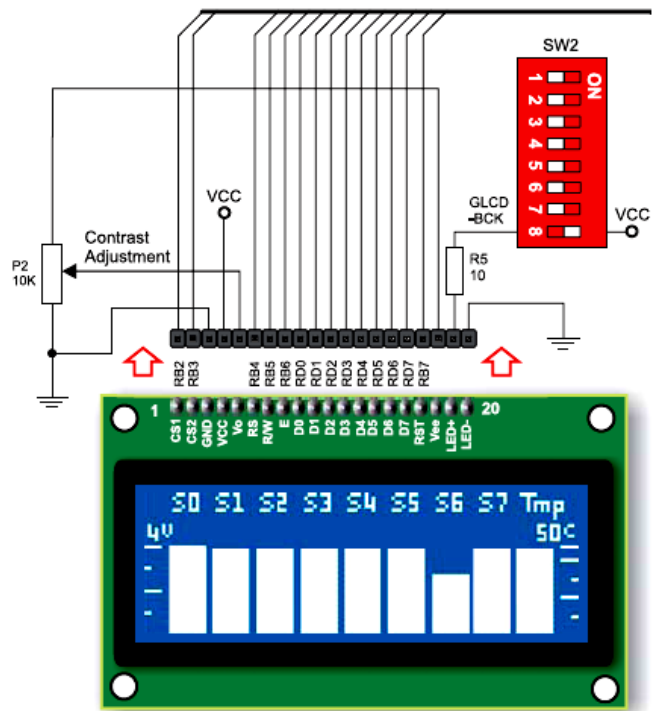
なお、PORTBの上位ビットだけは、ディップスイッチSW3によって、1ビット単位で個別にJP18で設定したプルアップ、ブルダウンの設定を有効/無効どちらにするか決めることができます。

これは、PORTBにはADコンバーターのアナログ電圧入力機能がアサインされているためで、PICD-32MXのボード上⑪のADコンバータ評価用ボリュームを使用する関係上、ビット単位で設定ができるよう設計されています。ADコンバーターを使用する場合には、使用するピンのスイッチをOFF側にセットします。

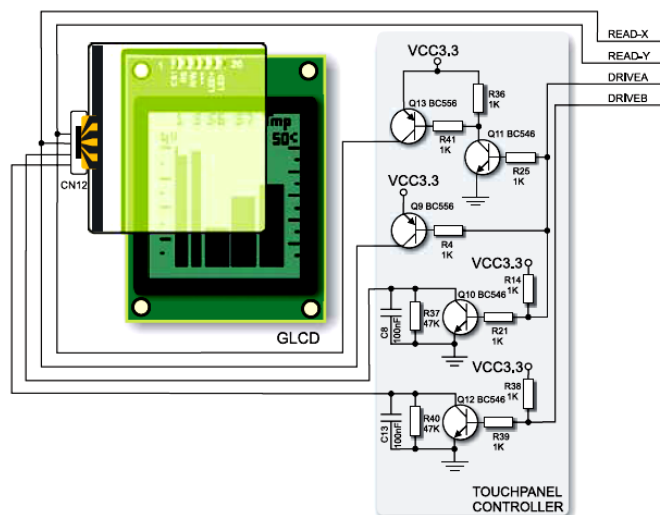
⑥グラフィックLCD用ピン4線式タッチパネル接続用コネクタ

128×64ドットのグラフィックLCDを接続するためのソケットです。また、キャラクタLCDを8ビットモードで使用する場合にもこのピンを使用します。

4線式抵抗膜方式のタッチパネル付きグラフィックLCDも装着できるよう、専用のコネクタも実装されています。

装着したGLCDのコントラストは、P2のボリュームで調整します。内部結線は下記の通りです。

GLCDのバックライトは、ディップスイッチSW2の8番スイッチでON/OFFを行うことができます。

タッチパネル用ソケットは、4線式抵抗膜方式専用で、当方からオプション品としてタッチパネル付きGLCDを販売しております。

タッチパネル用ソケットはディップスイッチSW2のスイッチ4〜7を介して本ボードのタッチパネルコントローラー回路部と結線されており、マイコン側からタッチ位置の検出をプログラムから行うことができます。結線は下図の通りです。

⑦100ピンMCUカードコネクタ

100ピンのMCUカードが装着できます。

装着時には、MCUカードの突起部分と、PICD-32MXボード上のシルク印刷の突起部分の方向が一致するように挿入してください。挿入方向を誤ると、機器を破損しますので十分ご注意ください。

なお、MCUカードの脱着時は必ずPICD-32MXの電源を切断した状態で行ってください。

⑧全I/Oピン接続ヘッダピン群

MCUカードに搭載のマイコンの全I/Oピンが取り出せるヘッダピンです。ポート単位で8ビットずつ取り出せるようになっています。

各ヘッダピンの左側には、ポート単位で8ビットずつプルアップ及びプルダウンが設定できるジャンパーJ15～J25があります。

PORTBは、ADコンバータ電圧入力用のボリュームが接続されている関係上、ディップスイッチSW3にて、上位8ビット(RB15～RB8)は1ビット単位でJ18の設定が有効か無効かに設定できるようになっています。SW3をOFFにすると、そのピンはオープンとなります。

このヘッダピン群に外部回路を接続する場合で、各ピンをデジタルI/Oとして使用したい場合には、使用するポートのLEDを物理的に切り離す必要があります。SW1及びSW2の設定により、使用するポートのLEDを切り離してご使用ください。

⑨周辺回路等設定スイッチ

SW3のディップスイッチは、PORTBの上位8ビット(RB15～RB8)に対して、J18で設定したプルアップ/プルダウンの結線を有効にするか、無効にするかを設定します。ADコンバーターを使用する場合には、アナログ入力となるピンのスイッチをOFFにして使用します。

SW4は、PICD-32MXに搭載の各種周辺回路と、マイコンのI/Oピンを物理的に接続するか、しないかを設定するスイッチです。

周辺回路を使用しない時は、電流が流れたり周辺回路からの影響を避けるためスイッチをOFFにして使用します。

⑩MPLAB-ICD2用モジュラージャック

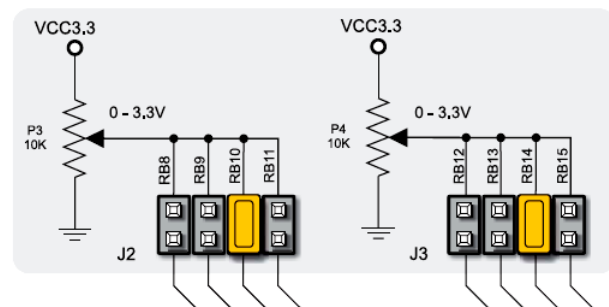
マイクロチップ社純正のICD、MPLAB-ICD2を接続できる6ピンのモジュラージャックです。MPLAB-ICD2をお持ちの場合、ここにICD2を接続すると、PICD-32MXをターゲットボードとして使用することができます。

詳しくは、本書4ページの「MPLAB-ICD2を使用する場合」の項をお読みください。

※MPLAB-ICD3は本ボードでは仕様上ご使用頂けません。

⑪ADコンバーター評価用ボリューム

ADコンバータ評価用のボリュームが2系統搭載されています。

J2及びJ3でボリュームを接続するI/Oピンを設定します。

ボリュームの電圧は、0V～3.3Vまでの範囲で変化します。

ADコンバータを使用する場合には、必ずADコンバーターに使用するピンのプルアップ・プルダウンの設定はOFFにしなければなりません。ディップスイッチSW3にて、ADコンバータとして使用するピンの設定をOFFにしてご利用ください。

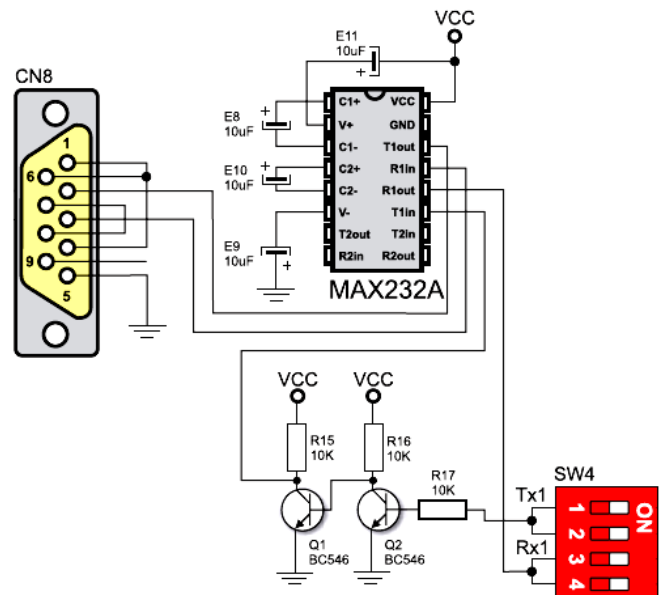
⑫2系統RS232Cポート

レベル変換IC搭載のRS232Cポートを2系統搭載しています。

左側のコネクタ(CN8)は、"A系統"です。

右側のコネクタ(CN7)は、"B系統"です。

A系統・B系統それぞれのTXピン、RXピンはディップスイッチSW4によって物理的にマイコンのI/Oピンと接続するか、切り離すかを選択できます。スイッチをON側にセットすると、配線が接続されます。

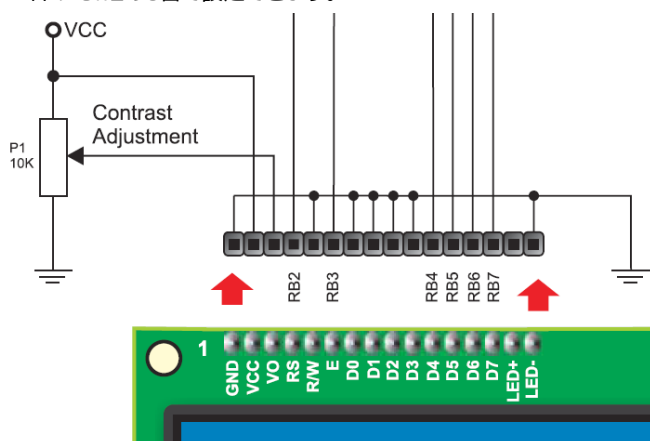
両系統のRX線及びTX線はそれぞれ2つのピンから選べますので、ご使用になるデバイスの仕様やプログラムの内容に合わせて適切に設定してご使用ください。

A系統の配線図を上図に示します。

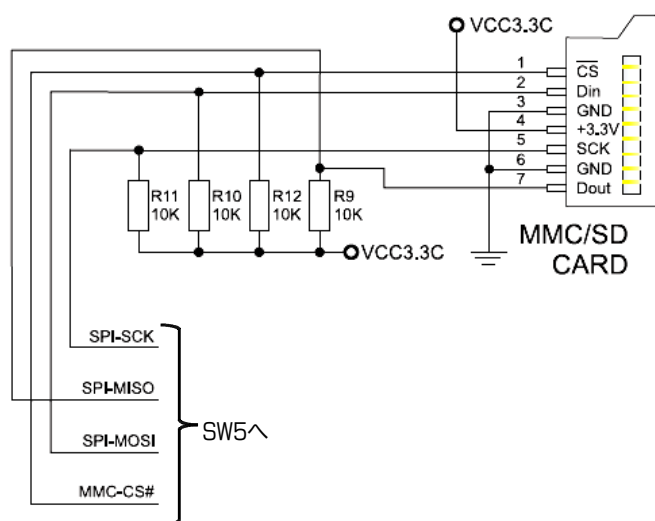
なおCTS線、RTS線は短絡されています。

⑭16文字×2行液晶ディスプレイ(4ビットモード)

16文字×2行のキャラクタLCDが装着できます。
LCDは、4ビットモードで駆動します。LCDのデータ線は上位4ビットを使用し下記のような配線になっています。

LCD側	マイコン側
RSピン	RB2
R/Wピン	GND(Wモード)
Eピン	RB3
D4ピン	RB4
D5ピン	RB5
D6ピン	RB6
D7ピン	RB7

LCDのコントラストは、P1ボリュームで調節できます。またバックライトはSW2の8番で設定できます。

⑯MMC/SDカードスロット

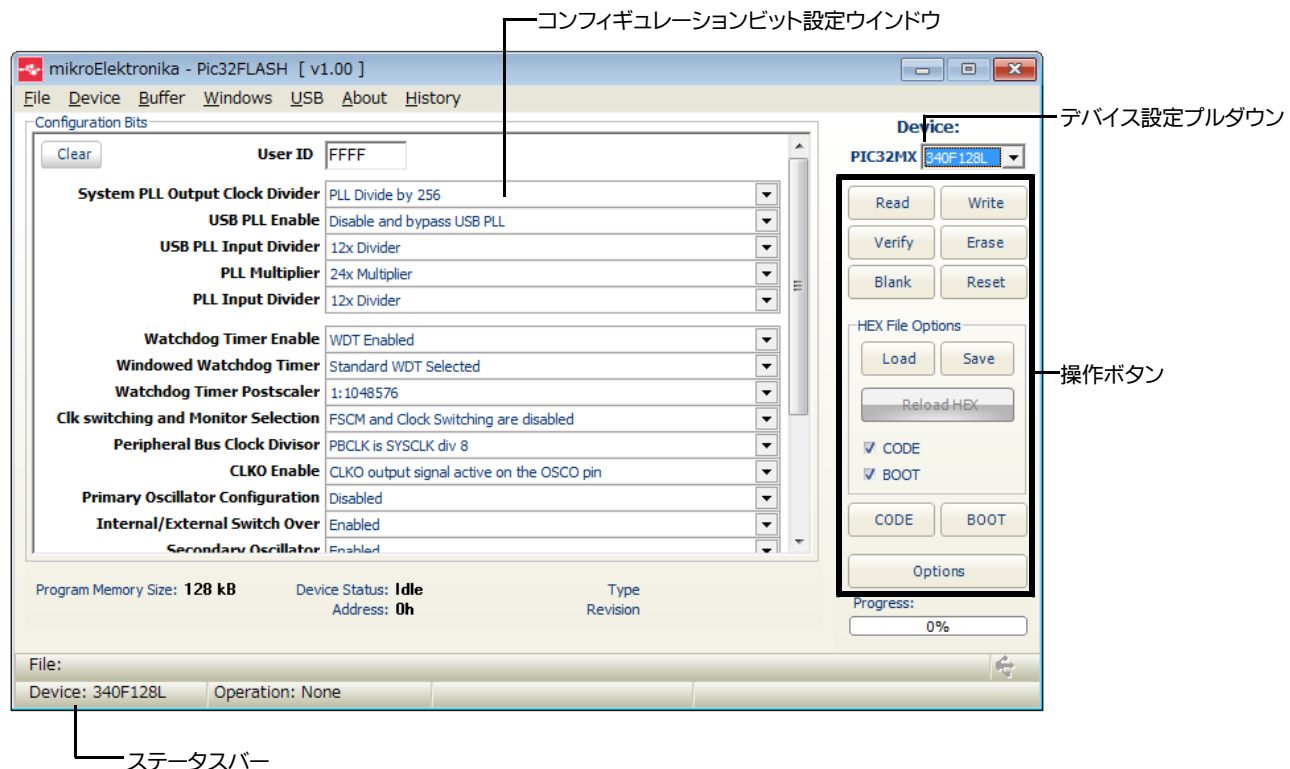
MMC又はMMCと電気的な互換性のあるSDカードを、SPIモードで制御してアクセスの実験ができます。
メモリーカードのCSピン、SDIピン、SCKピン、SDOピンはそれぞれ10KΩの抵抗でプルアップされ、ディップスイッチSW4を通してマイコンに接続されています。それぞれの信号線をマイコンと接続するかどうかをSW5によってON/OFFできます。

なおメモリーカード用の電源はPICD-32MX内で作っていますので、装着すればそのまま使用できます。

書き込みソフトウエアの使用方法

付属のUSB2.0マイコンライターの使い方を紹介します。
マイコンライターは、MPLAB等で作成したHEXファイルをPICマイコンへ書き込む際に使用するソフトウエアです。

■マイコンプログラマーの起動

Windowsのスタートメニュー又はデスクトップに作成されたショートカットから起動できます。
スタートメニューから起動する場合には、
"スタート"→"プログラム"→"Mikroelektronika"→"PIC32FLASH Programmer"→"PIC32FLASH"の順でクリックして起動します。

■画面の概要

■コンフィギュレーションビット設定ウインドウ

PIC32MXシリーズのコンフィギュレーションビットの設定を行います。選択したデバイスによって、コンフィギュレーションビットの内容が異なるため、表示される内容も変わります。
コンフィギュレーションビットは、PICマイコンを動作させるために必要な最も基本的な設定で、HEXファイルを書き込む際にのみ設定が可能です。このコンフィギュレーションビットの設定が正しくないと、プログラムの内容が正しくてもプログラムは動作しません。

PIC32MXでは、デバイスが高機能化し従来の8ビットのPICマイコンと比べ、コンフィギュレーションビットの設定内容が格段に多くなり複雑になりました。
設定内容はデバイス毎に異なっていますので、ご使用に際しては必ず使用してしてるPICマイコンのデータシートをご覧の上正しく設定してください。

■デバイス設定プルダウン

MCUカードソケットに装着されているデバイスの種類を選択します。

■操作ボタン

Readボタン‥	PICマイコンからHEXファイルを読み込みます
Writeボタン‥	HEXファイルをPICマイコンに書き込みます
Verifyボタン‥	書き込まれたHEXファイルの内容が正しいか、整合性を確認します
Eraseボタン‥	PICマイコンのプログラムメモリーを消去します
Blankボタン‥	プログラムメモリーが空か確認します
Resetボタン‥	マイコンライターをリセットします
Loadボタン‥	パソコンに保存されているHEXファイルを読み込むときにクリックします
Saveボタン‥	読み込んだHEXファイルを保存します
ReloadHEXボタン‥	既に読み込んでいるHEXファイルと同じファイルを再度読み込み直します
CODE及びBootボタン‥	現在読み込まれているHEXファイルの内容を表示します
Optionボタン‥	オプション設定のダイアログを表示します

■ステータスバー

現在ソフトウエアに読み込まれているHEXファイルのファイル名や、PICD-32MXの動作状況などの情報が表示されます。

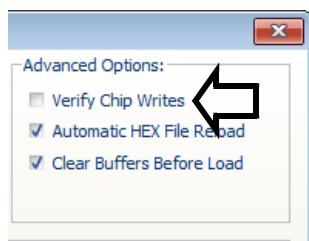
書き込みソフトウエアの初期設定と書き込み方法

■初期設定

PICFLASH32プログラマーでは、デフォルト設定(インストール直後の設定)でHEXファイル書き込みの際に、ベリファイ処理が実行されるように設定されています。ベリファイは、書き込んだHEXファイルを一度読み出して、照合することで書き込みが正しく行われたかどうかを確認する機能です。

しかしPIC32MXシリーズの場合、フラッシュメモリーのサイズが大きいためベリファイ処理を実行すると、照合作業に時間がかかり作業効率が落ちてしまいます。よってここではベリファイ処理を行わないように設定します。

1 "Option"ボタンを押します。ダイアログが表示されます。

2 "Advanced Options"の下にある"Verify Chip Writes"のチェックを外します。

3 "OK"ボタンを押して閉じます。

■HEXファイルの書き込み方法

次の順番でHEXファイルを書き込みます。

1 デバイス設定プルダウンより、書き込むデバイスを選択します。

2 書き込むHEXファイルを読み込みます。
操作ボタンの"Load"ボタンを押します。ファイルを開くダイアログが表示されますので、ファイルを選択します。
ファイルが読み込まれるとステータスバーに、ファイル名が表示されます。

※同じファイルを再度ロードする場合には、いちいち上記のようにファイルを指定しなくても、"Reload HEX"ボタンを押すことで再読込ができます。同じファイルを何度もデバッグして書き込む場合などに便利です。

3 続いてコンフィギュレーションレジスタの設定を行います。
コンフィギュレーションレジスタは、プログラムを書き込む時にだけしか設定できないPICマイコン全体の動作や基本的な設定などを行う特殊なレジスタです。
コンフィギュレーションレジスタの内容はデバイスに搭載される機能により変わるため、設定項目もデバイスにより様々です。
使用するデバイスのデータシートに詳しい内容が記載されていますので使用に際しては必ずデータシートをご参照ください。

4 設定が完了したら書き込みを行います。
PICD-32MX本体の"USB LINK"の黄LEDが点灯していることを確認してから操作ボタンの"Write"ボタンをクリックします。

■その他の機能

・Read機能

→PICマイコンからデータを読み込みます。
読み込んだデータは、"Code"ボタンで閲覧できます。またメニューバーの"File"→"Save HEX"で保存できます。

・Verify機能

→現在読み込まれているHEXファイルの内容と、PICに書き込まれているプログラムの内容が一致しているか検証します。

・Blankチェック機能

→現在装着されているデバイスのプログラムメモリがブランク(空)かチェックします。

・Erase機能

→現在装着されているデバイスのプログラムメモリの内容を消去します。

主な仕様

電源電圧:	USBバスパワー給電時 DC5V ACアダプタ給電時 DC9V〜12V
給電方法:	USBバスパワー又はACアダプタ
USB規格:	Ver,2.0対応
対応OS:	Windows98SE/ME/2000/XP/VISTA
対応デバイス:	PIC32MXシリーズ
生産国:	セルビア/中国/日本

サポート情報

PICD-32MXのサポートを行っております。以下のいずれかの方法でご質問をお寄せください。

■FAX番号　03-3700-3548
■電子メール　support@microtechnica.net

なお、MPLAB及びMPLAB C32についてのご質問は、当方では受け付けておりません。ご質問等は、マイクロチップ社にお問い合わせください。他社製品に関することや自作回路に関するご質問にはお答え致しかねますのであらかじめご了承ください。

ソフトウエアはアップグレードされることがあります。アップグレードされると対応デバイスが増えたりバグが修正されたりします。
アップグレードの情報などは当方のwebページにてお知らせ致しますので、下記webページを定期的にご確認ください。
http://www.microtechnica.net/manual/

PICマイコン用の統合開発環境,MPLAB及びMPLAB C32はマイクロチップ社製のソフトウエアです。バージョンアップ等の情報についてはマイクロチップ社のホームページより告知されますので,下記ページを定期的にご確認ください。
http://www.microchip.com/ (英語サイト)

マイクロテクニカ
〒158-0094 東京都世田谷区玉川1-3-10
TEL: 03-3700-3535 FAX: 03-3700-3548

PIC32MXマイコンの概要

PIC32MXシリーズは、米マイクロチップテクノロジー社が開発しているPICマイコンシリーズの1つで、CPUコアにMIPS社の32ビットコアを用いた新しいマイコンです。コアに他社製のものを採用することで、開発期間を短縮しました。

■PIC32MXシリーズの特徴

PIC32MXでは、32ビットマイコンとしての特徴をいかせるように様々な工夫がされています。プログラムメモリーも大容量化され、より大きなプログラムが実装できます。周辺機能も豊富に搭載されています。

動作速度の高速化も特徴の1つです。動作クロックは最大80MHzとなっており、1サイクルが1クロックで動作し最大80MIPS(Million Instructions Per Second＝1秒間に何百万個の命令が実行できるかという単位)で動作します。ただし、CPU本体がこれだけ高速に動作しても、フラッシュメモリーの動作速度はもっと遅いため、フラッシュメモリーからデータを読み出しながら実行するのではこの高速性能を生かし切れません。そこでPIC32MXでは、フラッシュメモリーとCPUの命令バスの途中にキャッシュメモリーを追加し、キャッシュメモリーからデータを取り出すことで80MHzの高速動作に追従できるよう設計されています。（ただし、キャッシュメモリーの動作はデフォルト設定では無効になっています。）

その他、割り込み処理はベクター方式となっており高速で動作させることができたり、演算機能が高速で動作したりと、従来のPICマイコンと比べ各種動作速度が格段に高速化されているのが特徴です。また、多くの周辺機能は8ビットや16ビットのPICマイコンと共通ですので、これまでにPICマイコンを使ったことがある技術者であれば、簡単に使うことができます。

■PIC32MX360F256Lの概要

PICD-32MXに標準で付属しているPIC32MX360F256Lの概要を紹介します。このデバイスはピン数100ピンのデバイスで、プログラムメモリーは256KBと大容量です。その他、12KBのブートフラッシュメモリーを搭載しています。データメモリーは32KBです。

周辺機能は、EUART/SPI/I2Cをそれぞれ2系統、分解能10ビットのADコンバータが16チャンネル、コンパレーターなどを搭載しています。

クロックはMCUカードに10MHzの水晶発振子がついており、これを利用することもできますし、内部に8MHzと32KHzの内蔵発振子を持っており、これを使うこともできます。

PIC32MXデバイスの電源電圧は、2.3V～3.6Vです。マイコン内部では、内蔵のレギュレターによって1.8Vに降圧されて電源が供給されています。

その他の詳細な機能についてはマイコンのデータシートをご覧ください。（データシートはCD-ROMに収録されています。）

MPLAB C32について

C言語は、ANSIという団体によって規格化された高級言語です。高級言語とはアセンブラ言語のように機械語に近いレベルの開発言語ではなく、より人間がわかりやすい言語体系の開発言語を指します。C言語を導入することでより簡単に、合理的にPICマイコンの開発ができるようになります。

C言語では様々なデータ型を扱えるほか、データを指し示すポインタや配列、関数などが使用でき、より高度なプログラムを記述することができます。

本PICD-32MXでは、マイクロチップ社からリリースされているMPLABという統合開発環境に、MPLAB C32というCコンパイラーを組み合わせて使用します。MPLAB C32は、製品版としても販売されていますが、"Evaluation Edition"というバージョンは無償で使用することができます。60日間は製品版と同等ですが、これを過ぎると最適化の機能が固定されます。しかし、よほど大きなプログラムを作らない限りはこの"Evaluation Edition"で十分です。

MPLAB C32は、マイクロチップ社のWEBサイトにユーザー登録すると、ダウンロードすることができます。チュートリアルを行う前にMPLABとMPLAC C32をインストールしておいてください。（本書の4ページ～5ページでその方法を解説しています。）

MPLAB C32、一般的なC言語としての機能の他にPICマイコンの機能をフルに使用できるよう様々な組み込み関数を搭載しています。組み込み関数を使用することで、様々な機能を実現できます。チュートリアルでは、そのうち一部の関数を使用して実際にプログラムを組みますが、すべてを紹介することは到底出来ません。組み込み関数についての詳細はMPLAB C32本体を解凍すると展開されるドキュメント、"32-bit-Peripheral-Library-Guide.pdf"に詳しく解説が記載されていますので、そちらをご覧ください。

チュートリアル①　～RB0の制御とボタン処理、変数の使い方～

では早速プログラムを作って開発を体験してみましょう。

■プログラムの概要

16ビットの変数を用意して、そこに入れた値を2進数でPORTBに出力するプログラムを作ってみましょう。変数を2つ用意して、外部のスイッチ入力によって、どちらの変数の値をRBに出力するのか指定できるようにしてみます。

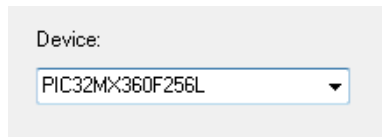
動作クロックは高速である必要がないので、10MHzで動作させることにします。

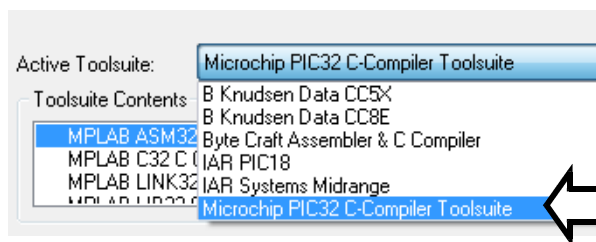
■プログラムを作成しましょう

1　MPLABを起動します。

2　MPLABでは、プログラムをプロジェクト単位で管理します。プロジェクトには、拡張子が.cのソースファイルを始めヘッダーファイルなど様々なファイルは1つのプロジェクトのもと管理する仕組みです。新しいプログラムを作る場合には必ず新しいプロジェクトを作成することをお奨めします。

最初にプロジェクトウィザードを使用して新規にプロジェクトを作成しましょう。MPLABのメニューバーから"Project"→"Project Wizard"をクリックします。

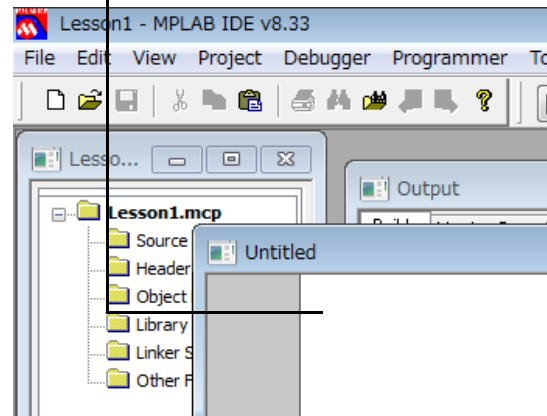
3　ウィザードが起動しますので、"次へ"を押して続行します。
次の画面でデバイスを選びます。
プルダウンから"PIC32MX360F256L"を選びます。

"次へ"をクリックして続行します。

4　コンパイラーの種類を設定します。"Active toolsuite"のプルダウンから"Microchip PIC32 C-Compiler Toolsuite"を選択します。

選択後、ブルダウンの下の"Toolsuite Contents"に実行ファイルの一覧が表示されます。この中に×印がすべて付いていないことを確認してください。もし×印が付いている場合には、正しくMPLAB C32がインストールされていない可能性があります。MPLAB C32が正しくインストールされているか確認してください。
もし手動でディレクトリを設定する場合には、"Browse"ボタンをクリックして実行ファイルのパスを設定してください。

"次へ"をクリックして続行します。

4　プロジェクトを作成するディレクトリを指定します。
"Create New Project File"の"Browse"ボタンをクリックしてディレクトリを指定します。

5　最後の"Setp Four"はそのまま"次へ"をクリックします。
"Summary"が表示されますので内容を確認して、"完了"をクリックしてウィザードを終了します。

6　続いてソースプログラムを作成します。
メニューバーの"File"→"New"をクリックします。
エディットウインドウが表示されます。

今はまだソースプログラムに名前が付いていませんのでタイトルは"Untitled"と表示されます。

7　エディットウインドウに下記のようにプログラムを記述します。

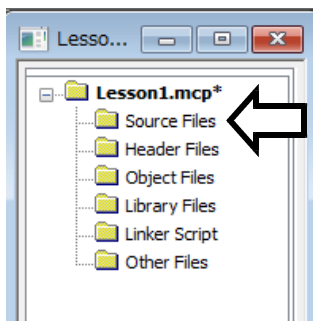
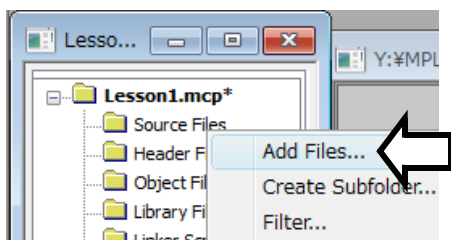
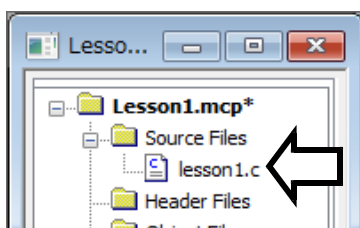
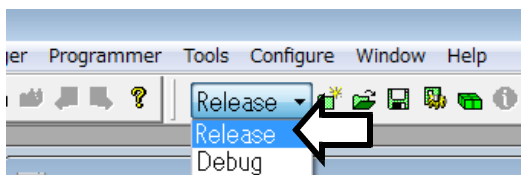
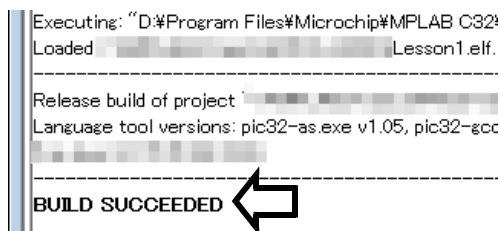
```
#include <p32xxxx.h>
#include <plib.h>

#pragma config FPLLMUL = MUL_16, FPLLIDIV = DIV_2
#pragma config FPLLODIV = DIV_1, FWDTEN = OFF
#pragma config POSCMOD = HS, FNOSC = PRIPLL
#pragma config FPBDIV = DIV_1

void main()
{
    unsigned int a , b ;

    AD1PCFG = 0xFFFF;
    TRISB = 0;
    TRISE = 0x3;
    PORTB = 0;
    PORTE = 0;
    a = 5200;
    b = 30000;

    while(1){
        if(PORTE==1){
            LATB = a;
        }
        if(PORTE==2){
            LATB = b;
        }
    }
}
```

記述したらミスがないかを確認してください。
特に; (セミコロン)の抜けがないかお確かめください。

■プログラムをビルドしましょう

プログラムをビルドすると、hexファイルが生成されます。このhexファイルをPICマイコンに書き込むとプログラムが動作します。ビルドをするためには、プロジェクトにビルドをするファイル類をすべて追加しておかなくてはなりません。下記の手順したがってビルドをしてみましょう。

1 最初に先ほど作成したソースプログラムに名前をつけて保存します。メニューバーの"File"→"Save As"をクリックします。ダイアログが表示されますので、ファイル名を付けて保存します。なおこのときの拡張子は必ず .c としてください。
ここでは例として、lesson1.cという名前とします。

2 続いてプロジェクトに1で保存したソースプログラムを追加します。MPLABに表示されているプロジェクトツリーの一番上にある"Source Files"を右クリックします。

3 メニューが表示されますので、"Add Files"をクリックします。

4 ファイルを開くダイアログが表示されますので、ここで先ほど手順1で保存した拡張子.cのファイルを選択して開きます。
ツリーにソースプログラムが追加されます。

5 これでビルドする準備ができました。ビルドするタイプを設定します。ツールバーにあるプルダウンから"Release"を選択します。

6 ビルドを行います。メニューバーの"Project"→"Build All"をクリックします。ビルドが開始されます。"Output"ウインドウが開き、内容が表示されます。画面一番下に"BUILD SUCCEEDED"と表示されればビルド成功です。

"BUILD FAILED"と表示された場合には、ビルドに失敗しています。エラーの原因が表示されますので修正します。
なお、"warning"は表示されていても問題ありません。"error"と表示された項目を修正してください。

hexファイルは、プロジェクトを作成したディレクトリの下に生成されています。

■PICD-32MXの設定をしましょう

プログラムを書き込む前に、PICD-32MXの設定を行う必要があります。PICD-32MXでは設定が正しくないとプログラムが期待したとおりに動作しなかったり予期せぬ動作をしてしまいます。プログラム書き込みの前に必ずプログラムの内容とボードの物理的な設定が正しいかどうかを確認する必要があります。

・SW1は"PORTB_L"と"PORTB_H"のスイッチ(3番と4番)の2つをONにして、他のスイッチはすべてOFF側にセットしてください。

・SW2・SW3・SW4はすべてOFF側に設定してください。

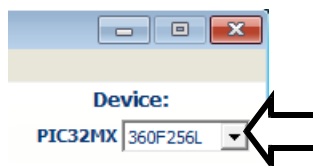
・J22のジャンパーを"Pull-Down"側にセットしてください。
　→これでPORTEのロジックは定常時はLレベルとなります。

・J1のジャンパーを"VCC3"側にセットしてください。
　→これでスイッチを押すと該当ピンに3.3Vの電圧が印加されます。

■プログラムを書き込み、動作を確認しましょう

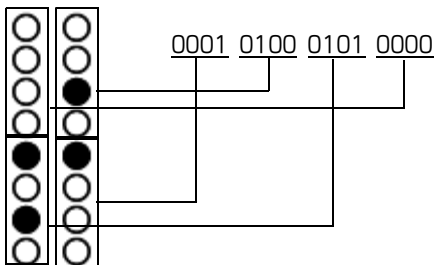
ボードの設定が完了したら、生成したhexファイルを書き込みソフトウエアでPICマイコンに書き込みます。書き込んだ直後からプログラムが動作します。下記の手順で書き込みます。

1 Windowsのスタートメニューから"プログラム"→"Mikroelektronika"→"PIC32FLASH Programmer"→"PIC32FLASH"の順でクリックします。 書き込みソフトウエア、PIC32FLASHが起動します。

2 最初にデバイスの種類を設定します。ウインドウ右上のデバイスプルダウンからPIC32MX360F256Lを選択してください。

3 先ほど生成したhexファイルを読み込みます。
"Load"ボタンをクリックします。ファイルを開くダイアログが表示されますので、先ほど作ったhexファイルを指定して開きます。
hexファイルは、プロジェクトを作成したディレクトリと同じディレクトリに作成されています。

4　一般的にhexファイルを書き込む前にはコンフィギュレーションビットの値を設定しますが、先ほどの例では重要な設定項目はソースプログラム内に記述していますので、hexファイルを読み込んだときに自動的にコンフィギュレーションビットの値は設定されます。
コンフィギュレーションビットの項目の中で、"Watchdog Timer Enable"の項目がを"WDT Disabled"に設定してください。

5　プログラムを書き込みますので、"Write"ボタンを押します。
書き込みが開始されます。(書き込みは数秒で終わります)

6　書き込みが完了したら、動作確認をしてみましょう。
まず、RE0のタクトスイッチを押します。RE0の時は、変数aの値、すなわち5200がPORTBに代入されますので、5200を2進数に変換すると、0001 0100 0101 0000 です。LEDを見てみましょう。
下図のように点灯しているはずです。

値の通りLEDが点灯していることがわかります。
同様にしてRE1のスイッチも押してみましょう。RE1を押すと変数bの値(30000)が代入されますので、0111 0101 0011 0000という値どおりLEDが点灯するはずです。

このほかに値を変えて試してみましょう。
なお最大値は16ビットの最大値 1111 1111 1111 1111 ですので、PORTBに0xFFFFを代入すると、LEDが全点灯となります。

■プログラムの解説

#include<p32xxxx.h>
#include<plib.h>
拡張子.hは、ヘッダーファイルといい各種定義などがひとまとまりになっているファイルです。PIC32MXを使う場合には共通のヘッダーファイルとしてp32xxxx.hというファイルがあります。このヘッダーファイルをインクルードしておくと各種レジスタや、ビット操作にレジスタの名称を使うことができます。
plib.hファイルは、周辺機能を使うためのライブラリです。
インクルードはプログラムの先頭で行うようにします。

#pragma config
コンフィギュレーションビットは、クロックの種類を設定したりクロック周波数の分周比を設定したり、ウォッチドッグタイマーの設定をしたりするPICマイコンの基本動作について設定を行うものです。
通常コンフィギュレーションビットは、hexファイルを書き込む時に書き込みソフトウエアで設定して書き込むものですが、いちいち書き込む際に設定をするのでは面倒ですし、間違ってしまうこともあります。そこでソースプログラムにあらかじめコンフィギュレーションビットの設定値を記述しておくことで、hexファイルを書き込みソフトウエアが読み込んだときに自動的にコンフィギュレーションビットの値を設定させるプラグマが用意されています。
※pragmaはプリプロセッサキーワードです。

"#pragma config"に続けてコンフィギュレーションビットのパラメーターと設定値を記述します。記述の仕方、項目の一覧はMPLAB C32のインストールされたディレクトリの下にあるDOCディレクトリに展開されるヘルプファイル "hlpPIC32MXConfigSet.chm"に型式ごとに記述がありますので、こちらを参考にしてください。

void main(){…}
main関数です。C言語では1つのプログラム内に必ず1つのmain関数が必要です。voidはmain関数の型で戻り値がない関数であることを宣言しています。{ }で囲まれた範囲がmain関数の範囲となります。

unsigned int a , b ;
符号なしのint型変数を宣言しています。C言語では文の最後には必ず ; (セミコロン)をつけます。
符号なしのint型の場合扱える値は、0～65535までです。

AD1PCFG = 0xFFFF;
AD1PCFGレジスタは、ADコンバータ用のアナログ電圧入力ピンを設定するレジスタです。該当のビットが0でアナログ入力、1でデジタルI/Oに設定されます。アナログ入力ピンはPORTBにアサインされていますので、ここでは、PORTBすべてのビットがデジタルI/Oになるよう0xFFFFを代入しています。

TRISB = 0 ;
TRISE = 0x3;
TRISレジスタは、ピンの入出力方向を設定するレジスタです。
該当のビットが0の場合は出力、1の場合は入力となります。本チュートリアルではRBは全ピン出力、REはRE0とRE1が入力なので0x3として設定しています。
C言語では16進数の値を表現する時は数値の前に0xを付けます。

PORTB = 0 ;
PORTBに0を代入しています。=(イコール1つ)は代入演算子で右辺の値を左辺に代入します。mikroCでは「レジスタ名=値」の書式でレジスタに値を代入できます。

a = 5200 ;
b = 30000 ;
int型変数に値を代入しています。

while(1){…}
while文は( )内の値が1又は真であれば{ }内の処理を繰り返す繰り返し文です。ここでは評価式を1としていますので永久に{ }内が実行されます。
C言語では処理を永久ループさせたい場合にはよくwhile文を使用します。

if (PORTE==1){ ;
if文を使用した入力判定部分です。
()内の評価式が真の時にその後ろに記述された文が実行されます。
==(イコール2つ)は比較演算子の「等しい時」を表す演算子です。よくある間違いとして=を1つしか記述しない場合があります。=が1つの場合には代入演算子となってしまい正しく動作しません。文法上は間違いではないのでエラーがでないため間違いを見逃してしまいプログラムが思うように動かない･･･ということがあります。

ここでは、「PORTBが1になった時」という条件判定をしています。PORTBが1になるとき・・というのはRB1のスイッチが押された時です。

LATB = a;

LATBレジスタに変数aの値を代入しています。
LATレジスタとは、出力ポートの制御に使うためのレジスタで、出力として使用する場合には、PORTレジスタと同じ動作をします。よって、この文は下記のように書き換えることもできます。

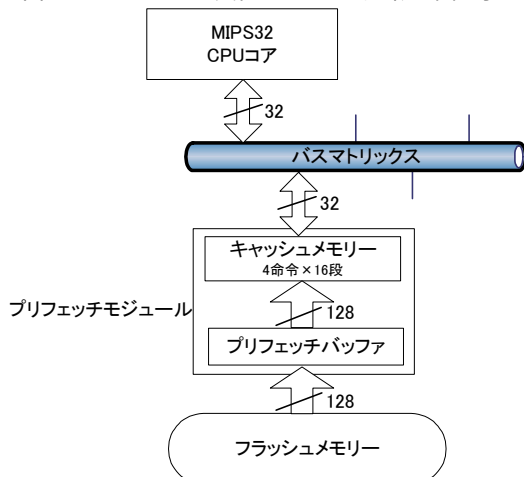
PORTB = a;

LATレジスタは出力として使用するときにはあまり意識しなくてもよいのですが、入力として使用するときには、注意が必要です。PORTレジスタは入力ピンの状態をそのまま読み込みますが、LATレジスタは、入力の時にピンの物理的な状態を読み込みません。LATレジスタはポートのLatchの状態を読み込むだけです。

ここまででチュートリアル①は終わりです。一通りハードウエアの使い方とソフトウエアでの開発を体験しました。ここまでの一連の操作は今後すべてのプログラム開発で共通ですので、しっかりと手順と方法を覚えておくことが重要です。これ以降のチュートリアルでは、プログラム本文とPICD-32MXボードの設定、プログラムの解説は掲載していますがソフトウエアの操作については解説していませんので、わからなくなった時はチュートリアル①を読み直してください。

チュートリアル② ～PIC32MXのスピードを体験する～

PIC32MXは、これまでの8ビットや16ビットのPICと異なり、1サイクルが1クロックとなっており、クロック周波数は最大80MHzまで出せる高速MPUとなりました。これは、コンピューターの性能指標の1つとして用いられるMillion Instructions Per Second(MIPS)という単位でいうところの80MIPSと表記できます。パソコン初期によく使われたインテル社の486DXというCPUは、54MIPSですので速度だけで見ればPIC32MXは初期のパソコンをしのぐ早さといえます。

しかし、1つ問題があります。せっかくCPUの速度は80MHzと高速なのにプログラムを記憶するフラッシュメモリーの速度は最高30MHzなので、その高速性能に追従できないのです。そこでPIC32MXでは、CPUとフラッシュメモリーの間にキャッシュメモリーを挟むことでその問題を解決しています。キャッシュメモリーは高速動作が可能なので、CPUの速度に間に合うのです。キャッシュメモリーは、4つの命令を16段格納することができ、最大64個の命令を格納できます。よって、あらかじめキャッシュメモリーに命令が入っていれば(プリフェッチといいます)CPUは80MHzという高速動作でも次々とキャッシュメモリーからデータを取り出してノーウェイトで実行できるのです。概念図を示します。

しかし、このプリフェッチモジュールのキャッシュメモリー機能は、電源投入時のデフォルト状態では、無効になっておりその効果を体験できません。ここでは、このキャッシュメモリー機能の有無によってどれだけ動作速度に差があるのかを実験してみることにします。

本チュートリアルで、外部10MHzの発振子から80MHzを生成する内蔵PLL機能の設定方法の他、プログラム内でディレイ(遅延)を作る方法を学習します。本チュートリアルで用いる方法で、プログラム内で用いる時間遅延を作ることができます。なお、本チュートリアルの実験にはオシロスコープをご用意ください。

■プログラムの作成

```
#include <p32xxxx.h>
#include <plib.h>

void main()
{
    SYSTEMConfigPerformance(80000000) ;

    AD1PCFG = 0xFFFF;
    TRISB = 0;

    while(1){
        LATB = 0x00;
        LATB = 0x01;
    }
}
```

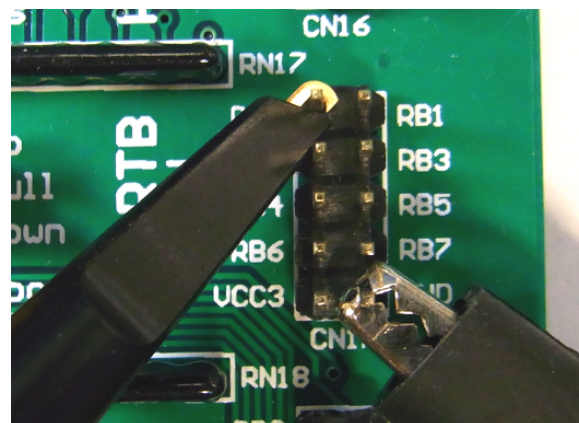
プログラムを記述したら、"Build All"でhexファイルを作ります。書き込みの前にボードの設定をしましょう。今回は手動でコンフィギュレーションビットの値を設定します。

■PICD-32MXの設定をしましょう

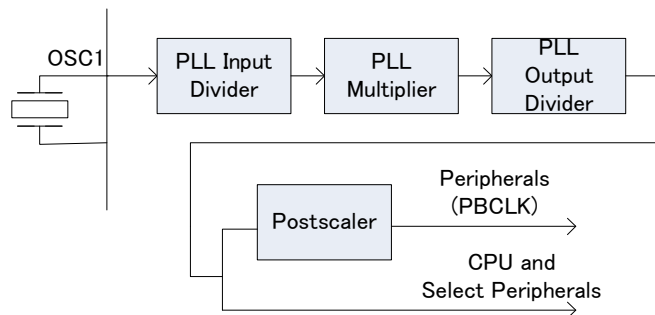
・SW1は"PORTB_L"のスイッチ(3番)だけをONにして、他のスイッチはすべてOFF側にセットしてください。

・SW2・SW3・SW4はすべてOFF側に設定してください。

■オシロスコープを接続しましょう

オシロスコープをお持ちでしたらRB0にプローブを取り付けましょう。PICD-32MXボードの右側にはすべてのI/Oと接続できるヘッダーピンがあります。RB0のピンにプローブを取り付けます。

RB0のパルスを観察します。電圧の振幅は3.3Vp-pですので、オシロスコープのレンジを調節してください。時間軸は20ns/div程度がよいでしょう。

■コンフィギュレーションビットを設定しプログラムを書き込みます

今回は、コンフィギュレーションビットの値を設定します。設定としてはクロック周波数を80MHzに内蔵PLLで逓倍することです。PIC32MXのオシレーター設定は複雑です。概念図を下図に示します。

外部の10MHzの水晶発振子のクロックは最初にPLL入力ディバイダに入ります。ここでクロックは分割され、周波数が低くなります。そのクロックはPLLに入り逓倍され、PLL出力ディバイダでさらに分割されます。そのクロックがCPUに供給される他、周辺用にはさらにポストスケーラーで分周したクロックが供給されます。

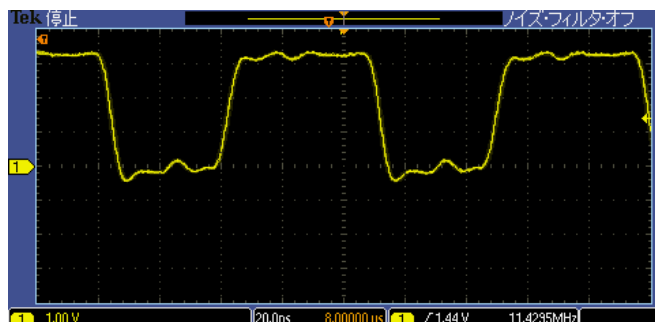
10MHzのクロックから80MHzのクロックを作るには、最初のPLL入力ディバイダでクロックは1/3にし、そのクロックをPLLで24倍します(10MHz÷3×24=80MHz)。PLL出力ディバイダでは1/1として分割せずそのままCPUに80MHzのクロックを供給することにします。

PIC32FLASHを起動します。最初にhexファイルを読み込みましょう。"Load"ボタンを押して生成したhexファイルを読み込みます。続いて、"Configuration Bits"の部分を設定していきましょう。下記のように設定してください。なお記載のない項目は変更せずそのままにしておきます。

・System PLL Output Clock Divider	PLL Divice by 1
・PLL Multiplier	24x Multiplier
・PLL Input Divider	3x Divider
・Watchdog Timer Enable	WDT Disabled
・Peripheral Bus Clock Divisor	PBCLK is SYSCLK div 1
・CLKO Enable	CLKO output Disabled
・Primary Oscillator Configuration	HS
・Oscillator Selection	Primary with PLL Module

設定ができたら"Write"ボタンをクリックして書き込みます。

■波形を観察してみましょう

オシロスコープの波形を観察してみましょう。下図は、オシロスコープの波形です。H期間がL期間より若干長い矩形波が観察できます。周波数は約11.43MHzとなります。

波形はやや歪んでいますが、L期間は約37.5ns、H期間は約50.0nsとなっています。なぜこのようなパルスとなるのでしょうか。このなぞの解決をするには、Cコンパイラーが作るアセンブラ言語のファイルを見てみる必要があります。C言語で書かれたプログラムはアセンブラ命令で構成されます。このアセンブラ命令の構成を見てみましょう。

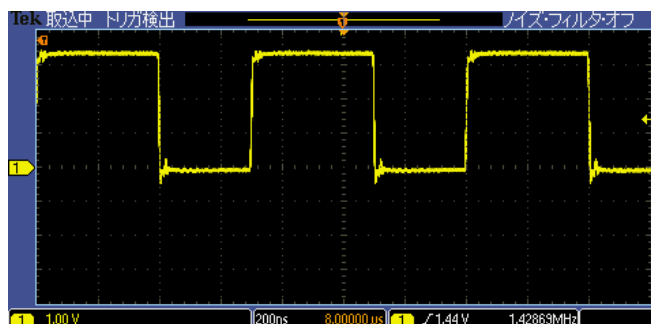
MPLABのメニューバーより"View"→"Disassembly Listing"をクリックします。スクロールバーを操作して下の方に移動すると、while文の内容を見ることができます。

```
19:
20:                          while(1)
21:                          {
22:                              LATB = 0x00;
9D0004CC  3C02BF88   lui         v0,0xbf88
9D0004D0  AC406060   sw          zero,24672(v0)
23:                              LATB = 0x01;
9D0004D4  3C03BF88   lui         v1,0xbf88
9D0004D8  24020001   addiu       v0,zero,1
9D0004DC  AC626060   sw          v0,24672(v1)
9D0004E0  1000FFFA   beq    zero,zero,0x9d0004cc
```

この内容を見るとLATB=0x00の後ろに2つのアセンブラ命令があります。80MHzで動作しているということは1サイクル12.5nsとなります。最初のLATB=0x00は3サイクルですので、12.5ns×3=37.5nsとなります。LATB=0x01の後ろは4つですので12.5ns×4=50nsとなり、オシロスコープの波形と一致します。全体では、7サイクルとなりますので、12.5ns×7=87.5nsとなり逆数にすると約11.43MHzとなります。

このようにC言語のプログラムを見ただけではわからない細かな動作内容もアセンブラの内容を見ることで理解することができます。ちなみにプログラム中の時間遅延はこのようにアセンブラ命令を繰り返し実行して時間稼ぎをして作り出すのが一般的です。

■キャッシュを使わない設定にして波形を観察してみましょう

今度はキャッシュメモリーを使用しない設定にして同じように波形を観察してみましょう。プログラム中の "SYSTEMConfigPerformance(80000000);" の部分の前に//(スラッシュ2つ)を入れてコメント行としてこの部分を実行しないようにします。そして再度Build Allでhexファイルを作り、先ほどと同様のコンフィギュレーションビットの値でプログラムを書き込み波形を観察してみましょう。

周波数は約1.43MHzとなりました。L期間は約300ns、H期間は400nsとなりました。先ほどの場合と比べると8倍遅くなっていることがわかります。"SYSTEMConfigPerformance();" という関数はキャッシュメモリーを有効にして80MHzの性能を出せるようにキャッシュメモリーを設定するためのものです。キャッシュメモリー使用の有無によって、8倍の時間差が生じることがわかりました。

同様にして、次はコンフィギュレーションビットの値を色々と変更して波形を観察してみてください。例えばPLLの倍率を24倍から15倍に変更してみましょう。クロックは50MHzとなりますので、動作も変わるはずです。オシロスコープで波形を観察しながら色々と変更してみてください。

チュートリアル③　～LCDの使用と、配列・ポインタ～

PICD-32MXに搭載の2行16文字のLCDに文字列を表示するプログラムを作成します。配列・ポインタの仕組みを学習します。なおLCDの駆動にはCD-ROMに収録されているPICD-32MX用のライブラリを使用します。

■プログラム作成前の準備

プログラム作成前にPICD-32MXのLCD駆動用ライブラリをプロジェクトを保存するディレクトリと同じディレクトリに入れておく必要があります。

CD-ROM内の"C Examples"フォルダを開きます。その中に"PICD-32MX Library"というフォルダがあるので開きます。"LCD_Lib1.c"というファイルがありますので、このファイルをこれからプロジェクトを作成するディレクトリと同じディレクトリにコピーしてください。

■プログラムの作成

PICD-32MXに装着されている2行16文字のLCDに文字列を表示します。C言語では、文字列を扱う場合配列を使用します。
本チュートリアルでは固定した文字列をLCDに表示させてみましょう。プログラムは次のように記述しましょう。

```
#include "plib.h"

#pragma config FNOSC = PRIPLL
#pragma config POSCMOD = HS
#pragma config FPLLIDIV = DIV_3
#pragma config FPLLMUL = MUL_24
#pragma config FPBDIV = DIV_1
#pragma config FPLLODIV = DIV_1
#pragma config FWDTEN = OFF
#pragma config OSCIOFNC = ON

#define  SYS_FREQ 80000000

#define  LCD_DATA LATB
#define  LCD_RS          LATBbits.LATB2
#define  LCD_E           LATBbits.LATB3

void  delay_us(int usec);
void  delay_ms(int msec);
void  lcd_out(char code, char flag);
void  lcd_data(char asci);
void  lcd_cmd(char cmd);
void  lcd_clear(void);
void  lcd_init(void);
void  lcd_str(const char *str);
void  MCUInit(void);

#include "LCD_Lib1.c"

int main(){
        MCUInit();
        char txt_a[] = "Hello";
        char txt_b[] = "World";
```

↓続きます

```
        lcd_init();
        lcd_clear();
        lcd_cmd(0x0C);
        lcd_cmd(0x80);
        lcd_str(txt_a);
        lcd_cmd(0xC0);
        lcd_str(txt_b);

        while(1){}
}
void MCUInit(){
        SYSTEMConfigPerformance(80000000);
        AD1PCFG = 0xFFFF;
        TRISB = 0;
        LATB = 0;
        delay_ms(100);
        return;
}
```

プログラムを記述した後はビルドをして、hexファイルを生成後、書き込みを実行してみましょう。LCDの1行目に"Hello"と表示され、2行目に"World"と表示されれば成功です。

■プログラムの解説

#define SYS_FREQ 80000000

#defineはプリプロセッサと呼ばれるコンパイル前にソースプログラムに対して行われる前処理のことです。プリプロセッサには他に、プログラムの先頭で記述している#includeなどがあります。
#defineはある文字列を別の文字列で置き換える時に使用します。(これを「マクロ定義」と呼びます)。
ここでは、SYS_FREQという文字列に80000000という数値を割り当てています。わかりやすい文字列を使用することでプログラムの可視性がよくなります。
"SYS_FREQ"は、インクルードする"LCD_Lib1.c"内で使用されているものでクロック周波数の値を定義する文字列です。

#define LCD_DATA LATB
#define LCD_RS LATBbits.LATB2
#define LCD_E LATBbits.LATB3

上記と同様マクロ定義で文字列に別の文字列を割り当てています。いずれも"LCD_Lib1.c"で使用されているもので、LCDのデータビットのポート名、RSピン及びEピンのビットの定義に使用しています。
MPLAB C32では、I/Oピンの指定には "LAT名bits.LATビット名"という規則になります。
PICD-32MXではLCDのデータピンはPORTB、RSピンはRB2、EピンはRB3に割り当てられていますので、上記のように定義します。

void delay_us(int usec);

main関数以外の関数をあらかじめ記述しておきます。
"void MCUInit();"以外はすべて"LCD_Lib1.c"内に記述されている関数です。

#include "LCD_Lib1.c"

PICD-32MXのLCD制御用に記述されたファイルです。インクルードすることで、LCDの各種制御ができるようになります。この中には、マイクロ秒単位の遅延をつくる"delay_us()"関数と、ミリ秒単位の遅延を作る"delay_ms()"関数が含まれています。いずれも精度は高くありませんので、おおよその遅延を作りたいときに使います。

MCUInit();
マイコンの初期化部分をまとめた関数を呼び出します。
main関数から別の関数を呼び出す場合には、関数名を記述するだけです。引数がある場合には(　)内に引数を記述します。

char txt_a[] = "Hello";
char型配列を宣言しています。char型は文字を扱う8ビットの型です。C言語では文字はASCIIコードにて扱われます。
通常配列を宣言する際には要素数(インデックス値)を[　]内に入力しますが、[　]内を空欄にすると自動的に要素数が設定されます。配列や変数を宣言する際に初期値を代入できますが、char型に限り文字列で初期化できます。"(ダブルクォーテーション)で囲むと文字列として認識されます。

lcd_init();
lcd_clear();
上記関数はLCDを使用する前に必ず実行します。
lcd_clear()関数はLCDの表示をクリアします。

lcd_cmd(0x0C);
lcd_cmd(0x80);
lcd_cmd()関数はLCDに制御コマンドを送信する関数です。
0x0Cはカーソル表示OFF、0x80はカーソル位置を1行目先頭に移動するコマンドです。

lcd_str(txt_a);
lcd_srt(*str)関数は指定した配列又はポインタの文字列をLCDに表示します。*strは、char型のポインタという意味です。ここでは、配列名を記述することで配列の0番目の要素のアドレス値を指定しています。

lcd_cmd(0xC0)
制御コマンド0xC0はカーソル位置を2行目先頭に移動させるコマンドです。

チュートリアル④　～TMR1割込を使用してLEDを点滅～

PIC32MXシリーズのデバイスには、TMR1という内蔵タイマーが搭載されています。TMR1は、16ビットのカウンターで指定したクロックに同期させてカウントを行わせることができます。

TMR1は、PR1レジスタの値と常に比較されるようになっており、TMR1の値と、PR1レジスタの値が一致した時、割込が発生する仕組みになっています。また割込発生後は、TMR1の値は0に初期化されます。

■PIC32MXシリーズの割込について
PIC32MXシリーズでは、割込の種類(何のイベントで発生した割込か)によって、ジャンプするベクタ(ジャンプベクタと呼びます)が異なるベクタ方式という割込を採用しています。8ビットのPICマイコンに比べ格段に構造が複雑になっています。
これにより、様々な割込に応じて異なる割込ルーチンを作成でき、より複雑な割込プログラムを作ることができます。

ジャンプベクタは、プログラムメモリーのアドレスに割り当てられています。割込が発生すると、その割込の種類によって、ジャンプベクタが変わります。どんな割込が発生した時、どのジャンプベクタへジャンプするのかは決められており、例えば今回使用するTMR1割込の場合には、ベクタ番号は4となります。このように割込の種類と、飛び先アドレスの一覧を主ベクタテーブル(IVT)と呼びます。割込ベクターテーブルの一覧は、PIC32MXシリーズのデータシートに記載がありますので、併せてご覧ください。

PIC32MXシリーズでは割込に順位が付きます。従来の8ビットのPICでは、割込は同時には1種類だけしか選択できませんでしたから、発生した割込の順位などは最初から必要ありませんでした。しかしPIC32MXでは複数の割込をベクタ方式で処理できるため、割込に優先順位を持たせてあります。
優先順位は、7段階で7が最も優先順位が高い割込になります。レベル7の割込の場合は、シャドーレジスタが使用されて、レジスタの退避と復旧が行われるようになっており、高速化を実現しています。

MPLAB C32には割込を使用するために利用できる関数が数多く用意されており、プログラムが作りやすくなっています。また割込発生時の処理は__ISRというマクロが用意されており、これを使うことでより簡単にプログラムの作成ができます。

■プログラムの作成
TMR1を利用してRB0に接続されたLEDを約1秒間隔で点滅させるプログラムを作成します。

```
#include ＜plib.h＞

#pragma config FPLLMUL = MUL_16
#pragma config FPLLIDIV = DIV_2
#pragma config FPLLODIV = DIV_1, FWDTEN = OFF
#pragma config POSCMOD = HS, FNOSC = PRIPLL
#pragma config FPBDIV = DIV_8

int main（void）
{
  AD1PCFG = 0xFFFF;
  TRISB = 0;

  OpenTimer1（T1_ON | T1_SOURCE_INT | T1_PS_1_256, 39062）;
  ConfigIntTimer1（T1_INT_ON | T1_INT_PRIOR_7）;
  INTEnableSystemMultiVectoredInt（）;

  while（1）;
}

void __ISR（_TIMER_1_VECTOR, ipl7） Timer1Handler（void）
{
        mT1ClearIntFlag（）;
        mPORTBToggleBits（BIT_0）;
}
```

プログラムが書き終わったら、PICD-32MXの設定を行います。今回はPORTBのLEDしか使用しませんので、ディップスイッチSW1の"PORTB_L"だけをONに設定して他のスイッチはすべてOFFに設定してください。設定が完了したらビルドしてhexファイルを書き込みます。

RB0のLEDが1秒間隔で点滅すれば成功です。

■プログラムの解説

このプログラムではTMR1の割込を使用して1秒間隔の点滅を作っています。TMR1は16ビットのカウンターです。本プログラムでは、TMR1のカウントアップのためのクロックには、ペリフェラルバスクロック(PBCLK)を使用します。

システムクロックは80MHzですので、それをPBCLKディバイダによって1/8にして10MHzとします。これをさらにTMR1のプリスケーラによって1/256にします。すると周波数は　80MHz÷8÷256 = 39.0625KHzとなります。すなわち25.6μ秒ごとにカウントアップされていくことになります。1秒を作りたいので、PR1レジスタに39062を代入しておけば、TMR1のカウンタが39062回目で割込が発生します。これはすなわち、25.6μ秒×39062=999.9872m秒ということで約1秒毎の割込となります。

TMR1は割込発生後は0にクリアーされます。

OpenTimer1(T1_ON | T1_SOURCE_INT | T1_PS_1_256, 39062);

OpenTimer1()関数はTMR1の動作モードを設定する関数です。

引数"T1_ON"でTMR1が有効になります。クロックのソースは、内蔵発振子を使用しますので、"T1_SOURCE_INT"を指定します。

プリスケーラは、分周比1:256にしますので、"T1_PS_256"にします。

最後にPR1レジスタに設定する上限値を記述します。ここでは39062を代入しています。

ConfigIntTimer1(T1_INT_ON | T1_INT_PRIOR_7);

ConfigIntTimer1()関数は、TMR1の割込条件を設定します。

"T1_INT_ON"で割込を許可します。

続いて、割り込みレベルを設定します。値は0～7です。7が最も割込順位が高くなります。

INTEnableSystemMultiVectoredInt();

ベクタ方式の割込を有効にしています。

void __ISR(_TIMER_1_VECTOR, ipl7) Timer1Handler(void)

__ISRは割り込み処理ハンドラ作成用のマクロです。

書式は下記の通りです。

```
void __ISR(vector, iplx)  XXXHandler(void){
        ハンドラの処理
        mXXXClearIntFlag();
}
```

このうち、vectorで指定する部分はベクター番号か、あらかじめ決まっている文字列を入力します。TMR1の場合には上記のような文字列の記述でもいいですし、ベクター番号4を入力してもよいことになります。

XXXNameのXXX部分も文字列が切れられており、TMR1の場合には"Timer1"としなくてはなりません。

割込の順位はiplxのxの部分に値を記述します。ここでは最も順位の高い7を記述しました。

この記述にすることで、TMR1の割込が発生した時はこの関数にジャンプするということが明示的に記述できるわけです。

mT1ClearIntFlag();

割り込みルーチンでは必ず記述します。TMR1割込フラグをクリアーします。このフラグをクリアーしないと割込がずっと発生していることになってしまい、新たに割込が発生しなくなります。

割込発生フラグはソフトウエアで必ずクリアーします。

mPORTBToggleBits(BIT_0);

I/Oピンに関する関数です。

指定したビットの状態をトグル(切り替え)します。HならばLに、LならばHに切り替えます。これを割込発生毎に実行するため、RB0が点滅します。

チュートリアル⑤　～非同期式シリアル通信(UART)～

PIC32MXシリーズでは非同期式シリアル通信(UART)モジュールを2チャンネル搭載しており、ハードウエアによるシリアル通信が可能です。

本チュートリアルでは、UARTにてデータを送受信するプログラムを作ります。なお、シリアル通信の通信速度は115.2kbps、データ長8ビット、パリティなし、1ストップビットという標準的なプロトコルとします。

※本プログラムの動作確認のためには、パソコンのRS232CポートとPICD-32MXのRS232Cポートを全結線ストレートケーブルで接続する必要があります。

■プログラムの作成

UART1を利用し、文字"a"以外のデータを受信したら、そのまま受信したデータを送り返すプログラムを作ります(エコーバックといいます)。また文字"a"を受信した時は、"HELLO"という文字列を送信するようにします。

```
#include ＜plib.h＞

#pragma config FPLLMUL = MUL_16
#pragma config FPLLIDIV = DIV_2
#pragma config FPLLODIV = DIV_1, FWDTEN = OFF
#pragma config POSCMOD = HS, FNOSC = PRIPLL
#pragma config FPBDIV = DIV_8

int main(void)
{
    unsigned char data;
    unsigned char str[]="HELLO";
    OpenUART1(UART_EN, UART_RX_ENABLE |
    UART_TX_ENABLE, 42);

    while(1){
        while(!DataRdyUART1());
        data = (char)ReadUART1();
        if(data=='a'){
            while(BusyUART1());
            putsUART1(str);
        }else{
            while(BusyUART1());
            putcUART1(data);
        }
    }
    return 0;
}
```

※"OpenUART1(UART_EN, UART_RX_ENABLE | UART_TX_ENABLE, 42);　"の部分は紙面の関係で2行で記載されていますが実際には1行で記述してください。

■PICD-32MXボードの設定

このプログラムを正しく動作させるため、下記のようにPICD-32MXボードの各種設定をセットしてください。

・SW1、SW2、SW3はすべてOFF
・SW4は"RS232A"のRF3・RF2のスイッチをON、それ以外はOFF

■コンパイルと書き込みの実行

これまでのチュートリアルと同様、ソースプログラムをビルドして、HEXファイルをPICD-32MXのマイコンに書き込んでください。

■パソコン側の準備

このプログラムでは、RS232C通信を行います。パソコンのRS232Cポートと、PICD-32MXの"RS232A"とかかれたポート(左側のコネクタ)を全結線ストレートケーブルで接続してください。

RS232CのターミナルソフトはWindowsに標準で付属しているターミナルソフト"ハイパーターミナル"が使用できます。その他使い慣れているターミナルソフトがある場合にはそれをお使いください。

通信プロトコルは次のように設定してください。通信速度は115.2kbps、データ長8ビット、パリティなし、1ストップビット。

■動作確認

パソコン側から"a"以外の文字列を送信してください。送信した文字例がそのままエコーバックして返ってくることを確認します。続いて、文字"a"を送信してください。"HELLO"という文字列が返れば成功です。

■プログラムの解説

OpenUART1( );

UART1モジュールのモードを各種設定する関数です。
UART1を有効に設定し、送受信を有効にしています。
また通信速度の設定は下記の式で算出しています。
設定値=動作クロック÷(16×ボーレート)-1　左式に代入すると次のようになります。
80MHz÷(16×115.2kbps)-1=42.40・・・　よって小数点以下を切り捨てて42を指定します。

while(!DataRdyUART1());

DataRdyUART1( )関数はUART1モジュールで受信データがある場合には0、受信バッファにデータがある場合には、1を返します。よって、ここではデータを受信するまで待機します。
データがない場合には、while文で繰り返して受信待機します。関数の前の!は論理を反転させます。よって受信バッファにデータが到達すると値が0となりループから抜け出します。

data = (char)ReadUART1();

char型変数dataに受信したデータを代入します。
ReadUART1( )関数は受信データをバッファから取り出して、その内容を返します。(char)で明示的に、返すデータの型を指定しています。

if(data=='a'){

分岐命令で受信したデータを判定しています。C言語の場合、文字を表す場合にはシングルクォーテーション(')で囲みます。文字列の場合には、ダブルクォーテーション(")で囲みます。

while(BusyUART1());

BusyUART1( )関数は、UART1の送信ステータスを返す関数です。1でビジー状態、0でレディ状態を示します。ここでは送信が可能になるまでループを繰り返し待機します。

putsUART1(str);

putsUART1( )関数は文字列を送信する関数です。引数として文字列又は配列、ポインタを指定します。ここでは、unsigned char型の配列strを指定して"HELLO"という文字列を送信しています。

putcUART1(data);

putcUART1( )関数は、送信データを書き込む関数です。引数には変数を指定します。

チュートリアル⑥　～ADコンバーターの使い方～

PIC32MXは、10ビットのADコンバーターを16チャンネル搭載しています。変換レートが1000kspsと高速です。基準電圧としては、内部リファレンス(vdd又はvss)か、外部リファレンスかを設定できます。
ここでは、PICD-32MXに搭載のボリュームから印加される電圧値をADコンバーターで処理して、結果をLEDに表示する簡単なプログラムを作成してみましょう。

■プログラムの内容

AN8(RB8)に印加された電圧をADコンバーターに入力し、10ビットの値に変換します。その結果をPORTDのLEDに出力するプログラムを作ります。

■プログラムの作成

```
#include ＜plib.h＞

#pragma config FPLLMUL = MUL_16
#pragma config FPLLIDIV = DIV_2
#pragma config FPLLODIV = DIV_1, FWDTEN = OFF
#pragma config POSCMOD = HS, FNOSC = PRIPLL
#pragma config FPBDIV = DIV_8

unsigned int ad_res;
unsigned int offset;

int main（void）
{
   CloseADC10（）;
   #define PARAM1   ADC_MODULE_ON | ADC_FORMAT_INTG | ADC_CLK_AUTO | ADC_AUTO_SAMPLING_ON
   #define PARAM2   ADC_VREF_AVDD_AVSS | ADC_OFFSET_CAL_DISABLE | ADC_SCAN_OFF | ADC_SAMPLES_PER_INT_2 |
                    ADC_ALT_BUF_ON | ADC_ALT_INPUT_ON
   #define PARAM3   ADC_CONV_CLK_INTERNAL_RC | ADC_SAMPLE_TIME_15
   #define PARAM4   SKIP_SCAN_ALL
   #define PARAM5   ENABLE_AN8_ANA

   SetChanADC10（ADC_CH0_NEG_SAMPLEA_NVREF | ADC_CH0_POS_SAMPLEA_AN8 ）;
   OpenADC10（PARAM1, PARAM2, PARAM3, PARAM4, PARAM5）;

   TRISD = 0;
   EnableADC10（）;

   while （ ! mAD1GetIntFlag（） ） { }

   while （1）
      {
         offset = 8 * （（~ReadActiveBufferADC10（） & 0x01））;
         ad_res = ReadADC10（offset）;
         PORTD = ad_res;
      }
          return 0;
}
```

■PICD-32MXボードの設定

　LEDは、PORTDを使用しますので、SW1とSW2は"PORTD_L"と、"PORTD_H"と記載されたスイッチのみON側に設定し、その他のスイッチはすべてOFF側に設定します。SW3及びSW4もすべてOFF側に設定してください。

　ADコンバーターの入力は、AN8です。AN8はRB8にアサインされていますので、J2のジャンパーは"RB8"の部分をショートしてください。

　16文字×2行のLCDは、ボードから取り外してください。

■動作確認

　プログラムをビルドして、HEXファイルを書き込んでください。
P3のボリュームを左いっぱいに回すと、LEDはすべて消灯します。
P3のボリュームを右に回していくと、印加される電圧の増加に従い、PORTDのLEDが点灯し始めます。

　ADコンバーターの分解能は10ビットなので、RD0～RD9までが点灯します。右にいっぱい回すと、10ビットすべてのLEDが点灯します。

■プログラムの解説

CloseADC10();

　ADコンバーターモジュールの動作を停止させます。
　SetChanADC10( )関数を使用して設定を行う前に必ず一度ADコンバーターモジュールを停止させておく必要があります。

#define PARAM1～PARAM5

　OpenADC10( )関数でADコンバーターモジュールを動作させる時に必要なパラメーターをあらかじめ、#defineで文字列PARAM1～5に割り当てています。こうすることで、OpenADC10( )の( )内にパラメーターを記述していくより、プログラムの見通しが良くなります。
　ここでの設定内容は主に次の通りです。

- ・出力形式は整数型
- ・トリガモードとサンプリングは自動設定
- ・ADコンバーターの基準電圧はAVddとAVssに設定
- ・ADコンバーターのクロックは内蔵RC発信器を使用
- ・AN8のアナログ電圧入力を有効に設定

　なおいずれもパラメーターはORでつなげます。

SetChanADC10(....);

SetChanADC10( )関数は、サンプリングするチャンネルを設定します。ここでは、AN8を入力として設定しています。

OpenADC10(....);

OpenADC10( )関数は、ADコンバーターを有効にして動作させる関数です。AD1CON1レジスタ・AD1CON2レジスタ・AD1CON3レジスタの各種設定を引数として記述し、ORでつなげます。ここでは、#define PARAMxによって、パラメータを文字列に置き換えて記述しています。

EnableADC10();

ADコンバーターの動作を開始させます。

while (! mAD1GetIntFlag()) { }

mAD1GetIntFlag( )はマクロで、ADコンバーターが値を取得すると、1を返します。よって、値取得まで待機させます。!は、論理値を反転させる否定演算子です。初回にADコンバーターからデータを取得するとき、有効なデータがADコンバーターの結果格納レジスタに格納されるのを待機します。

offset = 8 * ((~ReadActiveBufferADC10() & 0x01));

ReadActiveBufferADC10()はマクロで、ADコンバーターによって値が格納されているバッファの内容を返します。

PIC32MXシリーズのADコンバーター用のバッファは、16ワードのメモリーになっており、ADC1BUF0～ADC1BUFFまであります。このうち、バッファの0x0～0x7に値が書き込まれると0が、0x8～0xFに書き込まれると1が返ります。

~(チルダ)はNOT演算子です。

MPLAB C32では、ADコンバーターの結果を取得するReadADC10( )関数は、バッファからデータを読み出しますので、どのバッファから読み出すのかをあらかじめ決める必要があります。ここで読み出すバッファの値を変数offsetに代入しています。

ad_res = ReadADC10(offset);

ReadADC10( )関数は、ADコンバーターによって格納された値をバッファから読み出す関数です。バッファはADC1BUF0～ADC1BUFFまであり、ここでは引数に0～15の値を指定します。

ここでは、先のoffset変数によってバッファの値を指定しています。

なお、この関数の引数に指定するのはバッファの番号であって、アナログ入力のチャンネルではないことに注意してください。